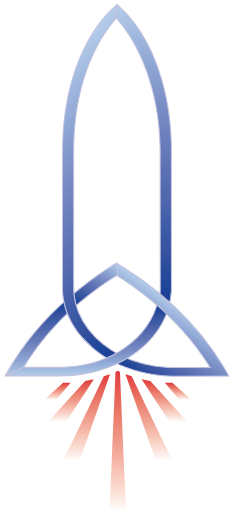
SpaceWire-2010

Proceedings of the 3rd International SpaceWire Conference

St. Petersburg 2010

Editors: Steve Parkes, Yuriy Sheynin, Martin Suess
Editorial Assistant: Lisa Rodway



Space
Technology
Centre
University of Dundee

SpaceWire-2010
Proceedings of International SpaceWire Conference
St. Petersburg 2010

ISBN: 978-0-9557196-2-2

Preface

These proceedings contain the papers presented at the 2010 International SpaceWire Conference, held in the House of Scientists, St. Petersburg, Russia, between 22 and 24 June, 2010. The International SpaceWire Conference aims to bring together SpaceWire product designers, hardware engineers, software engineers, system developers and mission specialists interested in and working with SpaceWire to share the latest ideas and developments related to SpaceWire technology.

SpaceWire links, nodes and networks are specified in the ECSS-E-ST-50-12C standard and additionally a number of higher layer protocols are defined in the ECSS-E-ST-50-51C, 52C and 53C standard documents, all published by the European Cooperation for Space Standardization - ECSS. Since the first publication of the SpaceWire standard in 2003, SpaceWire technology has developed into a true global standard for high speed data networks on board of spacecrafts. Today it is used in many space missions in Europe, USA, Japan, Russia and many other space-faring nations. But also the work on the SpaceWire and related standards has become a truly international undertaking. Representatives from space agencies, industry, institutes and universities all around the world are regularly discussing the evolution of the standardization during the SpaceWire working group meetings, which are organized by the European Space Agency. After the recent publication of the standards defining the Remote memory access and the CCSDS packet transfer protocols running over SpaceWire, the discussion in the SpaceWire working group currently concentrates on the definition of a protocol to use SpaceWire networks concurrently for command and control as well as for high throughput data transfers. Other topics are plug and play over SpaceWire, the definition of a SpaceWire backplane and the work on a SpaceWire handbook. Also the upcoming revision of the SpaceWire standard and SpaceFibre are discussed. All these topics can be found back in the papers published in these proceedings.

The objective of the conference is not only to bring this discussion on the evolution of the SpaceWire standard to a wider forum but also to give enough room for exchanging experiences made with the application of SpaceWire. Beyond SpaceWire standardization the sessions cover the presentation of missions using SpaceWire; new components, sensors and cables which support the SpaceWire standard; products supporting the SpaceWire standard including onboard equipment, instruments and related onboard software; methods and equipment to aid the test and verification of SpaceWire components, units and systems; and SpaceWire network architectures, their configuration, discovery, “plug and play” concepts as well as other higher level protocols and related hardware and software design issues.

We would like to acknowledge the support and hard work of the many individuals who made International SpaceWire Conference 2010 a reality. First, we thank the authors and the keynote speakers for their high-quality contributions. We express our gratitude to the Technical Committee for their gracious assistance in the review process. We thank all people supporting us at the St. Petersburg State University of Aerospace Instrumentation, the Space Technology Centre at the University of Dundee and the European Space Agency.

We acknowledge and are grateful for the generous financial support received from the Russian Federal Space Agency – Roscosmos and the European Space Agency.

The Conference Chairpersons,

Yuriy Sheinin, *St. Petersburg State University of Aerospace Instrumentation, Russia*
Steve Parkes, *Space Technology Centre at the University of Dundee, UK*
Martin Suess, *European Space Agency, The Netherlands*

Contents

Technical Committee

Philippe Armbruster - ESA, The Netherlands

Barry Cook - 4 Links Ltd., UK

Omar Emam - Astrium, UK

Wahida Gasti - ESA, The Netherlands

Alain Girard - Thales Alenia Space, France

Viacheslav Grishin- Submicron PLC, Russia

Sev Gunes-Lasnet - Astrium, France

Hiroki Hihara - NEC, Japan

Christophe Honvault - Astrium, France

Torbjorn Hult - RUAG Aerospace, Sweden

Jørgen Ilstad - ESA, The Netherlands

Gerald Kempf - RUAG Aerospace, Austria

Robert Klar - South West Research Institute, USA

Sergey Kochura - Reshetnev, Russia

Jim Lux - NASA JPL, USA

Peter Mendham - Scisys Ltd., UK

Masaharu Nomachi - University of Osaka, Japan

Olivier Notebaert - Astrium SAS, France

Steve Parkes - University of Dundee, Scotland, UK

Jaroslav Petrokovich - ELVEES, Russia

Manuel Prieto - Alcala University, Spain

Glenn Rakow - NASA GSFC, USA

Paul Rastetter - Astrium GmbH, Germany

Rashit Samitov - ENERGIA, Russia

Alan Senior - SEA, UK

Yuriy Sheynin - St. Petersburg State University of Aerospace Instrumentation, Russia

Tatiana Solokhina - ELVEES, Russia

Martin Suess - ESA, The Netherlands

Tadayuki Takahashi - JAXA, Japan

Raffaele Vitulli - ESA, The Netherlands

Takahiro Yamada - JAXA/ISAS, Japan

Programme Overview

Tuesday 22 June

08.30 - 10.00 - Registration (90 mins)

10:00-10:30 - Conference Opening / Keynote Presentation (30 mins)

- Conference Opening : Professor Yuriy Sheynin, SUAI
- Keynote Presentation: TBA

10.30 - 11.30 - Space Agency Presentations (60 mins)

- Roscosmos
- JAXA
- ESA
- NASA

11.45 - 13.00 - Standardisation (75 minutes)

14.30 - 15.15 - Test and Verification 1 (45 minutes)

15.45 - 17.15 - Test and Verification 2 (90 minutes)

Wednesday 23 June

09.00 - 10.40 - Components 1 (100 minutes)

11.10 - 13.00 - Components 2 (110 minutes)

14.30 - 16.00 - Poster Session/Exhibition (90 minutes)

16.00 - 16.45 - Components 3 (45 minutes)

16.45 - 17.50 - Onboard Equipment and Software (65 minutes)

Thursday 24 June

Thurs 09.00 - 10.45 - Networks and Protocols 1 (105 minutes)

Thurs 11.15 - 13.00 - Networks and Protocols 2 (105 minutes)

Thurs 14.30 - 15.50 - Missions and Applications 1 (80 minutes)

Thurs 16.15 - 17.55 - Missions and Applications 2 (100 minutes)

Programme is subject to change

Tuesday 22 June

SPACEWIRE IN PROSPECTIVE SPACE SYSTEMS AND INTERNATIONAL COLLABORATION

Session: Space Agency Presentations

Keynote Presentation

Sergey. A. Ponomarev

Deputy-Director of the Federal Space Agency of Russian Federation
(Roscosmos)

Complexity and scale of technology problems for modern and prospective Space exploration tasks and missions are so challenging that they could be efficiently solved only by joint efforts in international collaboration. A good example of efficient international collaboration is the International Space Station development, in which international participants independently designed their instruments and spacecraft avionics and integrated them with the standardized interface in the integral ISS data and control system. The standardized interface used was the MIL STD 1553B that had been developed in seventies of the last century.

Interfaces and protocols are the technology language for interaction of designers of units, systems and instruments. Interfaces development and standardization for spacecrafts and satellites in context of new tasks and missions, with new scope for industry production will give a way for new space technology development, which would provide assured construction of robust onboard systems in international collaboration.

I believe that the SpaceWire technology could settle these problems. The SpaceWire as the system integration and forming interface would enable building heterogeneous robust onboard system architectures that operate in real-time.

Unification and standardisation of interfaces is the key task for the general problem of spacecraft onboard systems and instruments unification. Roscosmos pay close attention to the unification problems. In the strategic target programme “Development of scientific and technology basis for spacecraft avionics and payload instruments design (2009-2020)” Roscosmos has formed the workgroup with the task of defining key interfaces for onboard systems, onboard computer system, to develop their architecture and pilot specimen design. With the concepts of integrated modular spacecraft avionics and onboard computing Roscosmos stimulates design of electronic components, VLSI and SoC, radiant tolerant processors, switches with integrated SpaceWire interfaces.

SpaceWire passed a long way in its evolution and recognition, from its initial development and publishing its first release in 2003 to present. The international SpaceWire WG works with support and active involvement of ESA, Roscosmos, NASA, and JAXA. SpaceWire developments are in the agenda of the EC–Russian Federation Dialog in Space area.

SpaceWire becomes the standard interface in national and international mission projects. Its further evolution, inclusion of Transport layer protocols and unified set of services in the SpaceWire family of standards give a way for wider unification in spacecraft onboard systems and payloads. It enabled conform SpaceWire network services to requirements and unified services that are specified for prospective spacecraft systems and payloads by the Consultative Committee for Space Data Systems (CCSDS).

With international experience of implementation and application of the SpaceWire technology – VLSI chips, instruments, space onboard data systems, tasks of further evolution of the SpaceWire technology are defined. New SpaceWire standard release should be prepared. Next generation SpaceWire standard development is to be launched for further increase of links throughput and length (“SpaceFibre”), for improved support of fault-tolerance and real-time operation of based on SpaceWire onboard systems (“SpaceWire-RT”). Roscosmos supports these activities and international collaboration in the developments.

In ESA and NASA cooperation some missions with SpaceWire based instruments have been implemented already. Russian developers of space onboard electronics (ELVEES, et al.) and onboard systems have designed and implemented some pilot specimens ready for flight testing and qualification. Instruments and units with SpaceWire interfaces are used in technology and mission satellites that have been designed by “Lavochkin” company, by RSC “Energia”, by “Reshetneva” ISS Company, by VNIIEM. SpaceWire technology is considered as the integrating concept and technology for space data system of the prospective piloted spacecraft, which is developed by the RSC “Energia”.

Roscosmos actively support activities in SpaceWire development and application, have great expectations on practice of its implementation and application in prospective national and international Space projects and missions.

Standardisation

SPACEWIRE NODES

Session: SpaceWire Standardisation

Long Paper

Martin Suess

ESA / European Space Research and Technology Centre, Noordwijk, The Netherlands

Albert Ferrer

Space Technology Centre, School of Computing, University of Dundee,

Dundee, Scotland, UK

E-mail: martin.suess@esa.int, aferrer@computing.dundee.ac.uk

ABSTRACT

The SpaceWire Standard ECSS-E-50-12C [1] is planned to undergo a revision. The main objective of this revision is to correct errors, remove ambiguities and to include some additional features which have been identified and agreed by the SpaceWire WG.

The title of the Standard [1] is "SpaceWire – Links, Nodes, Routers and Networks". While the Links and the Routers are well defined the experience from different implementations shows that the notion of what is comprised in a Node can differ. In conjunction with the new features that are going to be introduced during the revision of the standard to enable network discovery and PnP, the definition of a node in the SpaceWire network and the features it has to support need to be revised.

1 SPACEWIRE DEFINITIONS

As introduction some definitions of the current SpaceWire standard are reviewed.

1.1 NODE DEFINITION

The current standard [1] defines the nodes in section *10.4 SpaceWire nodes*:

"a. A SpaceWire node shall comprise one or more SpaceWire link interfaces (encoders-decoders) and an interface to the host system.

NOTE A SpaceWire node represents an interface between a SpaceWire network and an application system using the network services.

b. A SpaceWire node shall accept a stream of packets from the host system for transmission or provide a stream of packets to the host system after reception from the SpaceWire link, or do both."

SpaceWire nodes are defined here first of all and as connection point between applications and the network. Further nodes are characterised as sources and destination of packets. It is also clear that a SpaceWire node can comprise more than a single link interface.

1.2 ROUTER DEFINITION

The current standard [1] defines the nodes in chapter *10.2 SpaceWire routing switch*:

"a. A SpaceWire routing switch shall comprise a number of SpaceWire link interfaces (encoder-decoders) and a routing matrix.

NOTE The routing matrix enables the transfer of packets arriving at one link interface to another link interface on the routing switch, and the sending out from this link. Each link interface can be considered as comprising an input port (the link interface receiver) and an output port (the link interface transmitter).

b. A SpaceWire routing switch shall transfer packets from the input port of the switch where the packet arrives, to a particular output port determined by the packet destination address."

This chapter defines also that a routing switch must have an internal port with the address zero to access the configuration logic like e.g. the routing table. As safety measure this configuration port zero must only be accessible using path addressing. The maximum number of physical output ports of a router is limited to 31.

1.3 SPACEWIRE ADDRESSES

In chapter *10.2* it is also defined that packets in SpaceWire networks can be routed based on either path addresses or on logical addresses. A router always deletes the first byte of the path address (range 0 to 32) when routing a packet. When the packet reaches the destination node normally the complete path address has been deleted. A router can optionally also delete a logical addresses byte (range 32 to 254). This is done in gateways between addressing regions for regional addressing and may also be done on the final link to a node. The last point is never the case if the system is also compliant to [2] *ECSS-E-ST50-51C SpaceWire protocol identification*. There the target SpaceWire node always expects a logical address as the first byte in a packet followed by the protocol ID. There it is defined explicitly that one SpaceWire node may support several different logical addresses at the same time. These different logical addresses can for example be used to identify different virtual channels or applications in the node. The logical address 254 is the default logical address which shall be used where the target node has no other value specified or may be used when the logical address is unknown by the sending node.

1.4 TIME DISTRIBUTION

As specified in *8.12 System time distribution*, each node and router must contain one single six-bit time counter. This time counter is updated according to the Time-Codes received through any of the links or by the TICK_IN signal when the node is acting as

the time master in the network. Dependent on the value of this time counter compared to the value in the received time-code the time counter will be incremented and the TICK_OUT signal will be emitted. The consistent handling of the local time is one important aspect that is defining a node.

2 NETWORK DISCOVERY

In the frame of the discussion on Plug & Play for SpaceWire networks techniques for autonomous network discovery have been investigated. Network discovery is when a node can find out about the network topology and the functionality of the connected nodes by probing the network with interrogation packets. A lot of information on the network can be found by reading the configuration status accessible in a router through the configuration port. There the number of links of the router and their activity status should be accessible. These details can then be used to continue to discover the connected nodes and routers and in the end the complete network. One obvious problems is that when an interrogation packet is sent to an undiscovered device it is not known if this device is a node or a router. When trying to access a router with the default logical address 254 the packet will be normally spilled. On the other side, if using the path address 0 when accessing a node the same will happen. It became quickly clear that a unified method to access the configuration information of nodes and routers is needed for efficient network discovery. The currently planned revision of the SpaceWire standard gives the opportunity to introduce such modifications in the standard.

3 SPACEWIRE NODE MODIFICATION TO SUPPORT NETWORK DISCOVERY

After the discussions within the SpaceWire working group it has been decided to leave the definition of the router unchanged. Instead the definition of the SpaceWire node shall be adapted. Most importantly a configuration port which can be accessed using the path address zero should be also introduced as mandatory feature in the nodes to support network discovery. This gives the chance to review more general the definition of SpaceWire nodes and if this definition can be made more precise and needs to be updated. When looking at various SpaceWire implementations it can be perceived that currently different notions of what is a SpaceWire node exist. Sometimes a node is associated with a single logical address or even with an individual interface but often a wider definition is used.

3.1 NODE FEATURES

With the described modification, the concept of node is tied to a single configuration port which can be accessed from all SpaceWire links which belong to this node. In this port zero configuration space, among others, information about all links belonging to the node can be found. Similar important is that the node also contains only one single 6 bit time counter with a single time interface towards the host system. Time codes arriving through different SpaceWire links are handled equally and when the node is acting as the time master the time codes are sent out through all running SpaceWire links of the node.

Both the features are the same in nodes and routers. The main feature which distinguishes nodes from routers is that the nodes are the sources and destination of

packets and that they provide an interface to the host system which is using the network services.

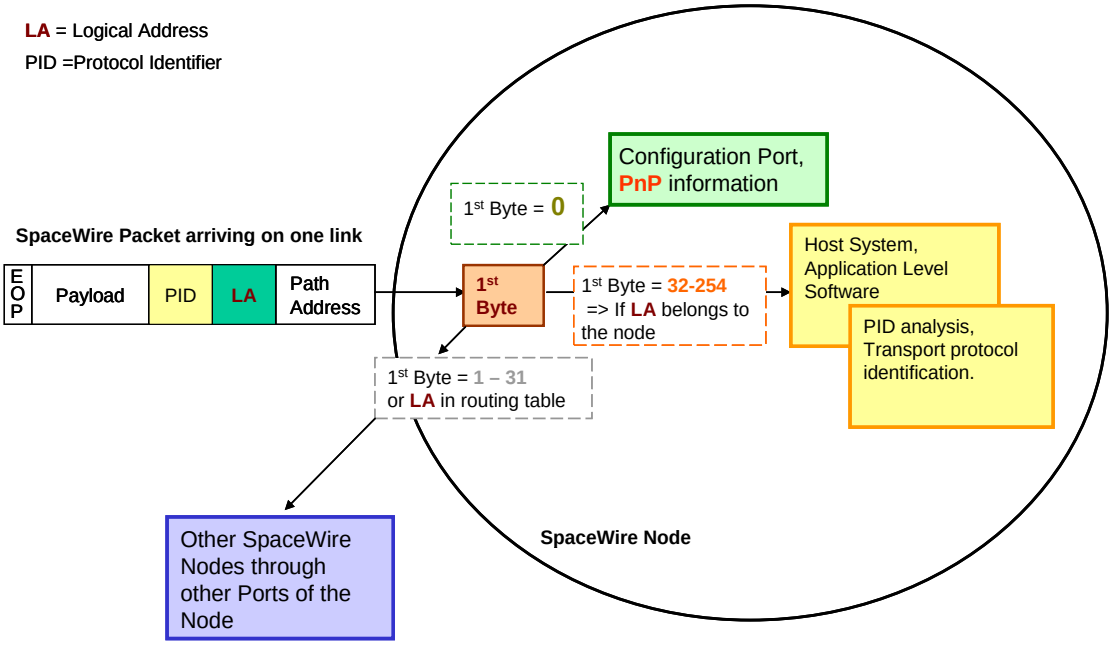


Figure 1: Handling of packet when arriving at a SpaceWire node

3.2 PACKET HANDLING BY THE NODE

The processing of a SpaceWire packet by a node following this definition is shown in Figure 1. The packet may have some leading bytes containing a path address. As specified in [2] this is followed by the logical address and the PID bytes and the payload of the packet. The node will start by analysing the first byte of the packet.

A. If the leading byte is a zero the packet will be routed to the configuration port for processing. The second byte would be expected to be one valid logical address of the node or the default logical address 254. The later is especially the case if a node is to be discovered and the logical address is not yet known by the sending node. The following handling of packet will be made in accordance with the Protocol Identifier (PID), which could for example indicate that it is a RMAP packet, a PnP packet or any other protocol supported by the node. It is important for network discovery that the node remembers the SpaceWire link through which it received the packet addressing the configuration port so that any reply to an interrogation packet is returned through the same link of the node.

B. If the leading byte corresponds to one of the Logical Addresses (LA) of the node the packet is forwarded to the host system. The PID and the rest of the packet may be analysed by hardware or software and may then be provided to the application level software for further processing.

C. One other possibility, which is not explicitly required or excluded by the current SpaceWire standard, is that the packet could be forwarded through on of the other SpaceWire links of the node. This forwarding could be based on a path address or a logical address defined in a routing table.

D. If the first byte does not correspond to any of the options mentioned before the packet is spilled.

4 ROUTING FUNCTION IN A NODE

Whether or not to include the optional routing function described under option C as part of the definition of the SpaceWire node has been controversially discussed during previous SpaceWire working group meetings.

For example the draft SpaceWire-PnP Protocol Definition [3] states that nodes are expected to have no routing function: “packets arriving at any port on a node will be consumed by the node.”

On the other hand there exist already some devices like the SMCS332SpW (AT7911E) which include such a routing function between the SpaceWire ports of the node. Similar, the Golden Gate ASIC developed by BAE [5], which can be used to connect up to four SpaceWire interfaces through a PCI bus to the host processor, also contains a routing function between the SpaceWire ports. There have been also a number of computer boards developed which make use of the SpW-10X router (AT7910E) to interface to the SpaceWire network. The SpW-10X provides two external ports that are effectively FIFO interfaces to inject and retrieve SpaceWire packets into and form the network. These examples make clear that nodes with integrated routing function are a concept which is actually widely used.

During a discussion it was proposed that these cases could be regarded as a node being attached to a router. Conceptually this could establish again the clear distinction between the routing and the network access point function in the Space Wire network. But as this connection is part of a SpaceWire network there should be one or several SpaceWire links between the router and this node. This is certainly not the case in the examples provided above and the reason is that implementing such a very short SpaceWire link is inefficient when the connection has to be made on a board or even in a chip. In addition it would also require a duplication of the configuration port zero. Conceptually this may be even be welcome but it will result in additional implementation effort. More significantly this separation of the node and the router function would also require a duplication of the local time counter. One belonging to the router and one belonging to the node. If the router and the node are attached to each other this duplication does not make sense.

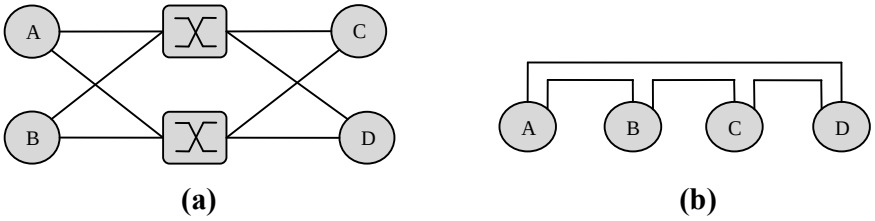


Figure 2: Example topologies to interconnect four nodes with redundancy, (a) Routing-centric topology with a redundant switch, (b) Ring topology using nodes with routing capabilities

Furthermore, the routing capability within a node allows useful network topologies as shown in Figure 2 (b). In some scenarios, a ring topology meets the requirements in

terms of bandwidth and redundancy without requiring external routing devices. The topology shown in Figure 2 (a) requires more harness, the powering of more devices and links that may not provide any advantage when for instance a simple chain of sensors is considered. On the contrary, the extra devices increase the complexity of failure cases and error recovery mechanisms. SpaceWire should not be constrained to certain network topologies and exclude other technologies which are widely used.

Finally it needs to mentioned that the presented node definition does allow the implementation of simple nodes with only a single link, nodes with several links without routing or nodes with several links with routing only between certain links. All these cases are a possible subset of the wider definition.

5 CONCLUSION

In this paper discussed the revised definition of a SpaceWire node that is needed to enable an efficient way of network discovery as part of PnP. For an interrogating packet the view of a node should be aligned with the one of a router by a mandatory introduction of a configuration port zero in every PnP enabled node as it is required in every router. The border of a node should be defined similar to the one of a router. All SpaceWire ports belonging to one node can access the same single configuration port zero and the time codes received through any link act on the same single time counter. The main distinction between nodes and routers is that a nodes provide an interface to the host system which is using the network service.

A node according to this definition may be also capable to forward a packet which it has received on one SpaceWire port through any other SpaceWire port of the node based on the physical or logical address contained in the first byte. As described this functionality is already implemented in a number of chips designed to provide SpaceWire interfaces for nodes.

It is suggested that this revised definition of SpaceWire nodes will be included in the next revision of the SpaceWire standard and that it will be reflected in the definition of the SpaceWire PnP protocol.

6 REFERENCES

1. “SpaceWire: Links, nodes, routers and networks”, ECSS-E-ST-50-12C, 31 July 2008.

2. “SpaceWire protocol identification”, ECSS-E-ST-50-51C, 5 February 2010.

3. Peter Mendham, “SpaceWire-PnP Protocol Definition”, Draft A Issue 2.1, 16 September 2009.

4. “SMCS332SpW User Manual”, SMCS_ASTD_UM_100, Issue: 1.5, 10 July 2007.

5. J. Marshall, “Evolution and Applications of System on a chip SpaceWire Components for Spaceborne Missions”, Proceedings International SpaceWire Conference 2008, 4-6 November 2008

6. SpW-10X SpaceWire Router User Manual”, UoD_SpW-10X_UserManual, Issue 3.3, 30 April 2008

STREAMING TRANSPORT PROTOCOL FOR SPACEWIRE NETWORKS

Session: SpaceWire Standardisation

Long Paper

Yuriy Sheynin, Elena Suvorova, Felix Schutenko

St. Petersburg State University of Aerospace Instrumentation
67, Bolshaya Morskaya st. 190 000, St. Petersburg RUSSIA

E-mail: sheynin@aanet.ru, suvorova@aanet.ru

Vladimir Goussev

SIC "ELVEES", Moscow
vgoussev@elvees.com

ABSTRACT

The basic SpaceWire Protocol Stack, standard ECSS-E-50-12C, covers a set of layers, from PHY to Network level. The Transport level framework is under standardization. It specifies general structure of the Transport PDU, claims that a variety of Transport protocols can be specified and work simultaneously in a SpaceWire interconnection. Transport layer protocols are in development with a couple of them standardised: RMAP, implementing remote memory access paradigm, and the CCSDS packet transfer protocol.

For Transport protocols we discuss a variety of choices between Connectionless (CL) and Connection-oriented (CO) protocols. The RMAP protocol is considered as a case study of a CL protocol. It is efficient for system administration, for setting/checking device parameters, for casual data polling. In regular and intensive data transfer the RMAP request/reply scheme could be of excess in overheads both in communications loads and operation overheads, non-consistent in the stream delivery to its consumer and in pumping data out from sources with limited buffering.

Many prospective applications to work over SpaceWire network interconnections operate with streaming data: data streams from high-rate sensors, ADCs, video streams input and output, etc. Some applications require support of multiple coherent data streams.

An outline for a new CO-type transport protocol – Streaming Transport Protocol (STP) is presented. We consider features that characterised streaming transport connection and consider the selected set of connection parameters, data packet parameters and additional data flow information.

The STP is implemented in our designs as a proprietary protocol. After its demonstration and trial it could be proposed for standardization by the SpaceWire community.

1. Introduction

According to the OSI 7-layer Reference Model, the transport layer is the lowest layer that operates on an end-to-end basis between two or more communicating hosts. The service of the transport layer use application entities. Communication between peer entities consists of an exchange of Protocol Data Units (PDUs). Application peers communicate using Application PDUs (APDUs),

while transport peers communicate using Transport PDUs (TPDUs). Interfaces between adjacent layers are provided with Service Access Points (SAP), by which the upper layer applies with its request to the lower one with data and control units that are the Service Data Units (SDU) of the layer in consideration. For the Transport layer it is the Transport Service Data Unit (TSDU) (used to be informally called a message), Figure 1.

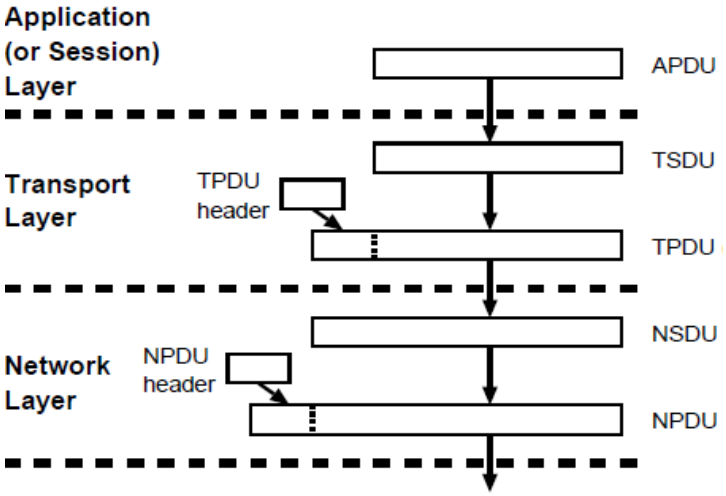


Figure 1.

The basic SpaceWire Protocol Stack, standard ECSS-E-50-12C, covers a set of layers, from PHY to Network level. The Transport level framework is officially added by the three new standards, [1, 2,3]. The ECSS-E-ST-50-51C introduces the Transport layer in the SpaceWire protocol stack and gives the SpaceWire transport protocol identification; the SpaceWire Transport layer is defined to be multiprotocol layer that supports simultaneous operation of multiple transport protocols running in a SpaceWire network. The next two standards specify the first two standardised transport protocols: the ECSS-E-ST-50-52C SpaceWire specifies the RMAP (Remote memory access protocol) transport protocol, the ECSS-E-ST-50-53C SpaceWire specifies the CCSDS packet transfer protocol transport protocol to run over SpaceWire interconnection.

Besides the standardised transport protocols the ECSS-E-ST-50-51C leaves a space for proprietary protocols also. With the 8-bit PIDs (Protocol Identifiers) coding the codes in the range 240 to 254 (0xF0 to 0xFE) could be used for particular non-standard protocols, one could develop and implement in its products. Reasons for developing new protocols could be specific requirements of particular projects and missions or lack of required for some transport services features and characteristics. In such cases a new protocol could be developed and implemented in addition to standardised ones as a proprietary transport protocol. With its implementation, which could be considered as proof of concept, and substantiation that similar features could not be covered with reasonable efficiency by the standardised protocols the new protocol cloud be standardised also by the SpaceWire WG.

Transport services can be divided into two types: connection-oriented and connectionless. A connection-oriented (CO) service provides for the establishment, maintenance, and termination of a logical connection between transport users. A transport service user generally performs three distinct phases of operation: connection establishment, data transfer, and connection termination. A connectionless (CL) service provides only one phase of operation: data transfer. A connectionless (CL) service provides no T-Connect and T-Disconnect primitives exchanged between a user sender and the transport sender, but gives only one phase of operation: data transfer.

As services are CO/CL specified, transport protocols could be classified CO or CL as well. The distinction depends on the establishment and maintenance of state information, a record of

characteristics and events related to the communication between the transport sender and receiver. A transport protocol is CO if state information is maintained between transport entities. If no state information is maintained at the transport sender and receiver, the protocol is CL. A CL protocol is based on individually self-contained PDUs often called datagrams that are exchanged independently. Each datagram contains all of the information that the receiving transport entity needs to interpret it.

The standardized Transport layers protocols RMAP and CCSDS PTP are connectionless protocols. More particular, the RMAP protocol could be classified as a transaction-oriented protocol, [4].Transaction-oriented protocols follows an asymmetrical model (i.e., client and server), short duration, low delay, few data TPDUs, and the need for no-duplicates service. Transaction-oriented protocols attempt to optimize the case where a user sender wishes to communicate a single APDU (called a request) to a user receiver, who then normally responds with a single APDU (called a response). Such a request/response pair is called a transaction. The RMAP protocol is efficient for system administration, for setting/checking device parameters, for casual data polling. In regular and intensive data transfer the RMAP request/reply scheme could be of excess in overheads both in communications loads and operation overheads, non-consistent in the stream delivery to its consumer and in pumping data out from sources with limited buffering, [5].

Many prospective applications to work over SpaceWire network interconnections operate with streaming data: data streams from high-rate sensors, ADCs, video streams input and output, etc. They have different features for which CL-class, transaction-oriented protocols like RMAP is not efficient. It motivated us in development of a new CO-type transport protocol for SpaceWire networks – the Streaming Transport Protocol (STP).

2. Streaming Transport Protocol (STP) features

The Streaming Transport Protocol (STP) is aimed for processing with stream-oriented information flow sources. Such types of sources could be found in many space systems and spacecraft payloads, e.g. ADC with high sampling rates, ADC with preprocessing, video cameras, SAR sensors, high-rate instrument sensors, etc. They have some general common features:

- Information flow is generated by the information source continuously.
- Information flow is a sequence of information chunks of fixed and the same length.
- Information chunks are generated by the source, may be periodically with some time interval.
- Information chunks length and generation time interval could change, but being changed they keep it operating for a long period.
- Corrupted and lost in transmission information chunks are not expected to be repeated; in most cases – could not be repeated by the source.
- The receiver cannot stop generation of the information flow by the source instantaneously.

Additional feature one can find in many applications is that a receiver, e.g. the payload data processing unit, quite often deals with a set of similar sources that form a set of data streams. Moreover, some applications require support of multiple coherent data streams and this feature is to be supported by the transport service also.

Such a set of features justifies development of the tailored for it protocol, we call the Streaming Transport Protocol, STP. The STP provides for applications the connection-oriented transport service that fits the target information flow features. Continuous data generation and sending with stable features do not require a mode control per data chunk. The logic link between a source and

the receiver and its mode of operation could be set once for a long period, thus omitting overheads for per PDU transmission and delivery mode control. The connection-oriented service and CO transport protocol look quite natural here.

The STP is developed as the CO transport protocol for regular data stream transfer from the source as the slave (with or without internal buffering). Interaction between the source and the recipient, the Transmitter and the Receiver is based on the establishment and maintenance a logical connection between these transport users. The master initiates establishment of the session, with setting logical connection – the transport channel, and setting its mode of operation and parameters. The session will be in operation until it will be terminated by the transport connection endpoint – the master.

Like the RMAP, the STP is an asymmetric protocol with the master and the slave(s). The master is the recipient and the slave is the source of the data stream to be transmitted. As distinct from RMAP, which requires a read command to be send to the slave to initialise transmission of data PDU from it, the STP initialises data transmission ones for a long period of operation. After it the source (the slave) will send data PDUs one by one in accordance with the set for the transport connection parameters. The source governs itself the moments of data PDUs transmission (on data availability, on its generation time interval, etc.), without per PDU requests from the master (receiver).

The STP forms its PDU form the SDU that is supplied by the Application layer through the STP SAP. The SDU is enveloped and transmitted in the single STP PDU; STP does not use packing/unpacking of an SDU into fragments. An SDU is reformatted into a PDU that is transferred to the SpaceWire Network level for transmission in a single packet. Transmitted by the transport connection PDUs have the fixed size that has been set in the transport connection establishment phase. Changes in any mode and parameters, the size included, in the STP could be done only by termination the connection and establishment of a new transport connection with modifies parameters.

With the STP idea of the host-receiver and slave-transmitter a STP transport connection is a point-to-point connection. A transport connection supports connection with a single slave and unicast service. However, it is considered that the master can have multiple transport connections with multiple slaves. Though they would be separate connections, they could be considered as correlated ones. From the master node (host) application point of view it could be a set of flowing together data streams, in some cases – coherent streams. Thus we have here an opposite to the multicast transmission case, so to say Inverse Multicast – a many-to-one transmission, [6]. To support of multiple coherent data streams feature the STP introduces a special field in data packets for coherence alignment of the incoming data streams in the receiver.

3. STP phases

3.1. Connection establishment

A Connection establishment launches the transport session between the host and a slave and builds the transport connection between them. The transport connection is asymmetric logical connection for transmission of packets from the source (slave) to the recipient (master). In the opposite direction the master sends control PDUs (commands) to the slave. On the transport connection (TC) establishment the TC parameters are set that would be in operation for the whole TC lifetime. The cost associated with the connection establishment would be amortized over a connection's lifetime.

Initiator of connection is the receiver (master). For the STP avoiding false connections is important, so 2-way or 3-way handshake mechanisms with explicit exchange of control TPDUs are needed. When the underlying network service provides a small degree of loss, a 2-way-handshake mechanism may be good enough to establish new connections without significant risk of false connections. The SpaceWire interconnections have rather low BER. However high robustness requirements for the space grade onboard interconnections shift the level of risk that is acceptable; it motivated us to move to the 3-way handshake. Three-phase protocol is used by the STP for connection establishment, Figure 2.

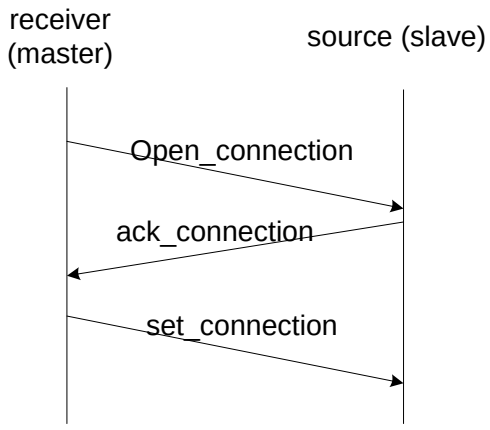


Figure 2.

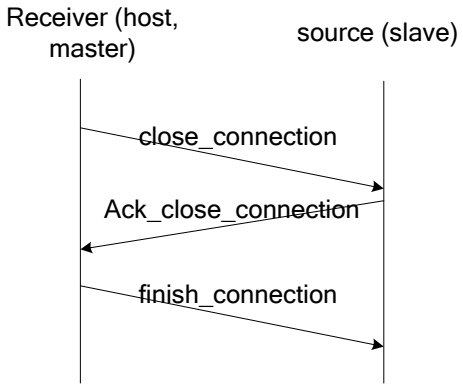


Figure 3.

The master sends an Open Connection to the slave, which responds with an Ack_Connection. The procedure is completed with a Set_Connection. No user data is carried on the connection establishment TPDUs. The 3-way-handshake is needed to prevent false connections that might result from delayed TPDUs.

3.2. Data transfer

Data PDUs could be transferred after the transport connection has been established. Permission for data transmission is sent by the receiver that should send a *Start_transfer* command to the receiver. A data PDU is generated by the slave and is sent to the master by the transport connection between the slave and the master.

Data transfer at the transport layer requires some flow control by which the recipient would not be flooded by the incoming PDSU flow and the underlying interconnection would not be blocked by packets that the recipient cannot intake. It can be done by preventing a transport sender from sending data for which there is no available buffer space at the transport receiver, or by preventing too much traffic in the underlying network. The SpaceWire interconnection doesn't have a standardised feature for preventing traffic overflow at the Network layer (though some implementations could have it). The STP Flow control mechanism uses the receiver crediting End-to-End Flow Control (E2E FC). The receiver (master) issues credits n in the number of packets it has buffer space for. The transmitter can send no more than the number of packets it has credits for. The packet size is defined in the transport connection parameters that are set in the Connection establishment phase and is known to both sides of the TC

The master can send the $n = 0$ that means the credit an unlimited number of packets. In fact, it is switching of the credit-based E2E FC. In many applications the receiver (the master) knows the PDU flow rate that could be generated by the source and is quite sure that could process the incoming data PDU flow in its regular mode of operation. The transmitter will send a packet after a packet by the TC without waiting for anything from the receiver to continue this process. The receiver shall receive and take them away from the TC. In case it cannot do it after some time, it

can stop the PDU flow by sending Stop control PDU to the transmitter. The transmitter should stop sending data PDU immediately as soon as it receives this control PDU. It is realised that a set of data PDUs could be left in the TC (in the underlying interconnection), which has been send in the interval between the moment when the receiver decided to stop transfer and the moment when the transmitter has received the Stop command. The receiver is obliged to take out all the left after its Stop command issue packets; the receiver can use them if it has buffer place for some of them or throw them away, these packets are considered to be lost.

The STP provides the ordered transport service. As an ordered service it preserves the user sender's submission order of data when delivering it to the user receiver (in-order delivery). It never occurs that a user sender submits two pieces of data, first A, then B, and A is delivered after B is delivered.

The STP considers ordering as providing a linear order of SDU generation events. For a SDU generation event in the source the strict order relation is defined to the events of all other PDUs' generations. The wall clock time of an event is not considered by the STP protocol FSM.

The STP understands that a basic SpaceWire interconnection does not guarantee in-order delivery of the sent packets. To control the in-order delivery of the STP PDUs and to reconstruct the initial PDUs order it includes the ordinal number of the SDU in the STP data packet format.

The STP has been developed in the general scope of the SpaceWire evolution, following its basics, compact implementation included. Thus the reordering of PDUs is limited by some number of *k* packets (a TC parameter). Outside the window of *k* packets a limited in-order delivery is provided, the reconstruction of the initial packet order at the receiver side is not guaranteed. Instead, the violating the order packets are discarded; their places in the ordered SDU sequence, which is returned to the upper layer, are filled by the default filler SDU (a TC parameter).

The STP provides a not guaranteed PDU delivery. An Error Control is provided, but corrupted or lost packets are not reconstructed or retransmitted. It corresponds to the nature of many steaming data applications. The receiver does not inform the transmitter of receive errors and do not request to resend PDUs. The source does not keep a sent by it PDU and do not retransmit it. However, the upper layer at the receiver side is informed about errors in the forwarded to it SDU flow.

3.3. Connection termination

Closing of the session and termination of the current transport connection with the slave is initiated by the master. For connection termination the STP uses a three-phased protocol that is illustrated by Figure 3. When the session and the TC are closed all its parameters are reset.

Two 2-way-handshakes are used, one for each direction of data flow. The master transport entity sends a *Close_connection* to its peer entity. The slave (transmitter) stops to send next data PDUs and then acknowledges the disconnect request by *Ack_close_connection*. The connection is terminated when all the incoming data flow is received by the master(receiver) – a sequence of PDUs in the TC finished by the *Ack_close_connection*. It ensures graceful TC termination, in normal operation no data in transit will be lost. Next the master confirms that it has received the acknowledgement and considers the TC is closed. After both sides have come to the "Closed" state for the TC, they become ready for establishment of another connection between them.

4. STP packet formats.

The STP uses three basic packet formats

- Packet-Command with parameters
- Packet-Command without parameters
- Data packet

4.1. STP Command packets

List of the STP commands (command PDUs) is presented by the Table 1.

Table 1. STP commands

Code	Command	With/without parameters
0000	open_connection	With parameters
0001	ack_connection	Without parameters
0010	set_connection	Without parameters
0011	close_connection	Without parameters
0100	ack_close_connection	Without parameters
0101	finish_connection	Without parameters
0110	start_transfer	Without parameters
0111	stop_transfer	Without parameters
1000	credit_transfer	Without parameters
1001 - 1111	Reserved	--

Commands without parameters are simple and compact (SpW header plus 5 bytes). In fact, commands with parameters are not used in the data transfer phase at all; all the possible for this phase commands are without parameters. The format of packet-commands without parameters is represented at the Figure 4.

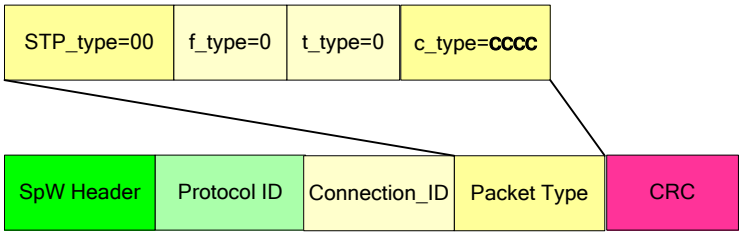


Figure 4

For the STP it is considered that the master can have multiple transport connections simultaneously; so the commands contain connection identification. Connection identifier is placed in Connection_ID field; its value could be from 0 to (2^{16} – 2), with the FFFF intended for special goals. Thus the STP master can have up to (2^{16} – 1) TCs with different slaves.

The Figure 5 represents general format of packet-command with parameters.

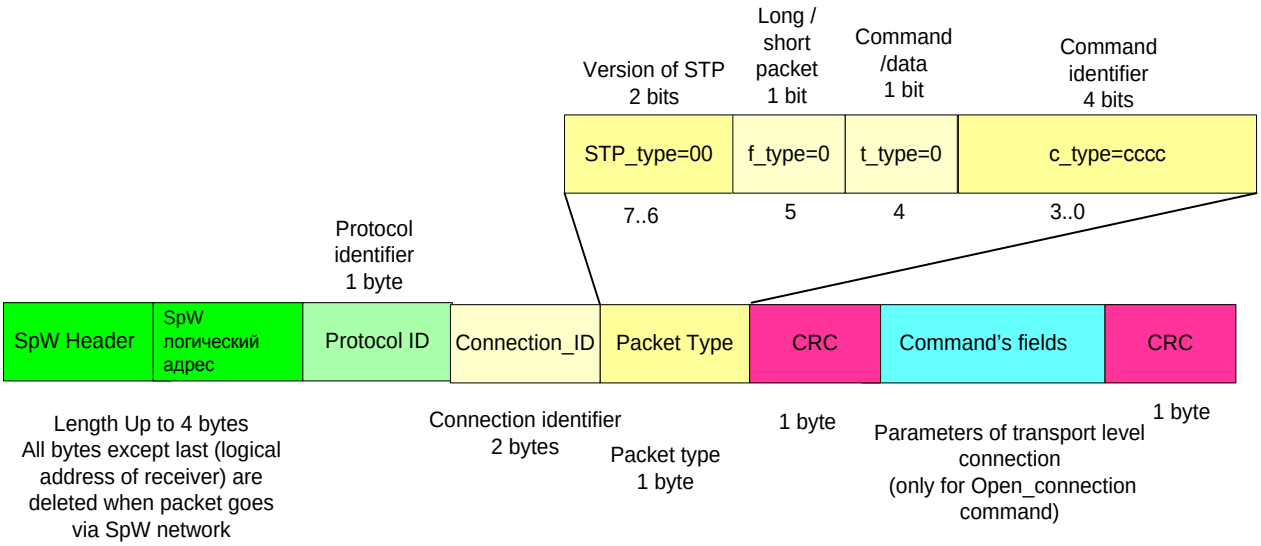


Figure 5

4.2. STP Data packets

Format of data packets represented at the Figure 6. The solid lines mark the boundaries of the 32-bit words (informative).

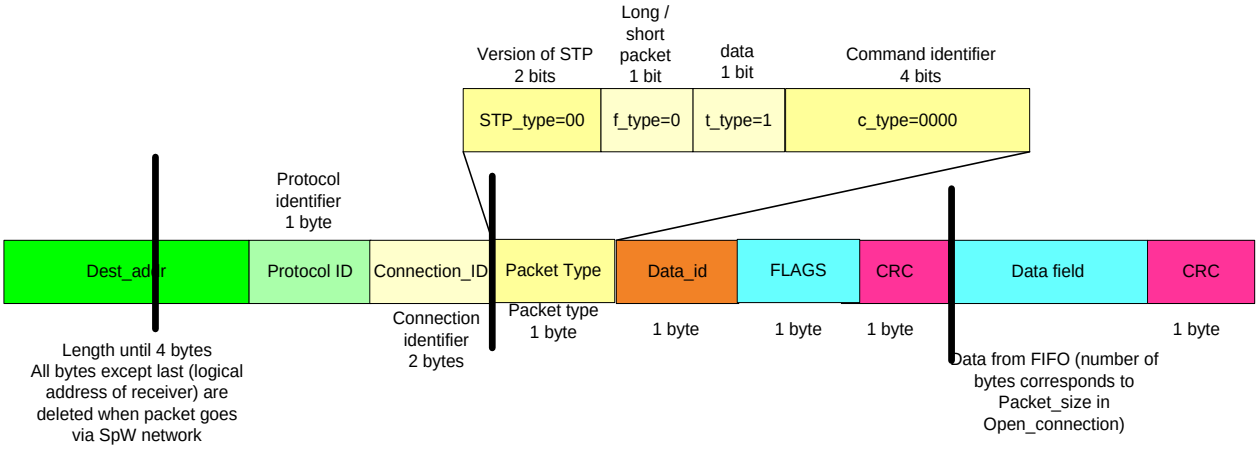


Figure 6

The *Data_ID* field (1 byte) is used for data identifier. Data identifier is generated by the source. Data identifier of every next data PDU is incremented by one; after 255 is the data identifier 0. If the transmitter received *Stop_transfer* command, after the following *Start_transfer* the Data_ID of first data packet will be 0.

The sFLAGs field is not specified in the STP; it could be used for flags that are generated and processed at the Application level. For instance, it could be used by the Application level protocols for its piggybacked control information transfer.

Conclusion

The Streaming Transport Protocols covers the streaming data transfer over the basic SpaceWire networks, which are not supported efficiently by the standardized Transport layer protocols.

The STP could be implemented over the basic SpaceWire interconnections with existing switching routers. Like the RMAP it could be implemented in nodes, e.g. processor-based nodes, in peripheral microcontroller nodes, in software. However, a hardware implementation of the STP is also quite feasible and can give better throughput and latency characteristics. Different STP implementation profiles could be specified also around its core functionality giving more cost-efficient specialization for particular applications with strict resource constraints. The STP is implemented in our designs as a proprietary protocol. After its demonstration and trial it could be proposed for standardization by the SpaceWire community.

Further developments could cover alternative, from the master to slaves, and bi-directional data streams transfer that are not covered by the current STP specification version. More interesting features, e.g. real-time coherent data streams stamping and correlation, could be built in STP with rely on the future SpaceWire 2.

References

1 ECSS-E-ST-50-51C SpaceWire protocol identification. ECSS Secretariat, ESA-ESTEC 2010, 5 February.

2 ECSS-E-ST-50-52C SpaceWire - Remote memory access protocol. ECSS Secretariat, ESA-ESTEC 2010, 5 February.

3 ECSS-E-ST-50-53C SpaceWire CCSDS packet transfer protocol. ECSS Secretariat, ESA-ESTEC 2010, 5 February.

4 Braden, R. Extending TCP for transactions – concepts. RFC 1379 (1992. Nov.).

5 Shutenko F., Suvorova E., Yablokov E., Gillet M. Low Power Protocols Development and Implementation. Proceedings of 6-th Seminar of Finish-Russian University Cooperation in Telecommunications, FRUCT 2009, pp.113-121

6 Gorbachev S., Sheynin Yu. Transport layer protocol for transputer networks. “Nauchnoe Priborostroenie” (Scientific Instrumentation), 1994, No 3-4. pp. 55-60 (in Russian).

SPACEWIRE-D: DETERMINISTIC DATA DELIVERY WITH SPACEWIRE

Session: SpaceWire Standardisation

Long Paper

Steve Parkes, Albert Ferrer,

Space Technology Centre, School of Computing, University of Dundee, Dundee, UK

E-mail: sparkes@computing.dundee.ac.uk

Stuart Mills, Alex Mason

STAR-Dundee Ltd, c/o School of Computing, University of Dundee, Dundee, UK

Email: stuart@star-dundee.com, alex@star-dundee.com

ABSTRACT

How can data be delivered deterministically over an existing SpaceWire network using existing SpaceWire components i.e. with no modification to the SpaceWire interface or router hardware? The approach explored in this paper is the sharing of system bandwidth using time-division multiplexing. The paper explores deterministic delivery over SpaceWire networks, describes how this can be achieved with current SpaceWire nodes and routers, provides corresponding theoretical performance estimates, and reports on a practical demonstration of SpaceWire-D.

1 INTRODUCTION

SpaceWire provides a versatile network architecture which is ideal for many space applications [1]. It has been widely used for payload data-handling on more than 30 space missions. SpaceWire is ideal for data-handling applications but does not address avionics and other applications where responsiveness, robustness, determinism and durability are essential requirements. There is a need for a spacecraft avionics network technology which combines the key features of SpaceWire with the quality of service requirements of real-time avionics applications. One critical requirement for avionics applications is deterministic delivery of information.

2 THE PROBLEM WITH DETERMINISTIC DATA DELIVERY OVER SPACEWIRE

Deterministic data delivery is the delivery of data within predetermined time-constraints: not too early and not too late. The essential consideration is a priori knowledge of when data will be delivered and the level of uncertainty in that knowledge. SpaceWire-D (where D stands for determinism) is a protocol that provides deterministic delivery over a SpaceWire network [2]. SpaceWire-D delivers data within predetermined time constraints.

SpaceWire is an asynchronous network which uses wormhole routing. The leading byte(s) on a SpaceWire packet determines the route through the network using path or

logical addressing. When the start of a packet arrives at a router it is switched to the required output port straightaway, provided that the required output port is not already being used to transfer another packet. Storing and forwarding of packets is not used by SpaceWire routers. This reduces the amount of buffer memory required in the routing switches. A disadvantage arises when the output port is already being used to transfer another packet and the packet has to wait for the output port to become free. The packet will be left strung out across the network from the blockage back to the source of the packet. It will prevent any other packet being transferred across the links that the packet is occupying.

This gives rise to the main difficulty in providing deterministic data delivery over SpaceWire: access to the SpaceWire network has to be controlled to avoid conflicting use of network resources. For example, two packets that need to travel down the same link at the same time will result in one packet having to wait while the other is transferred, leading to possible temporary blockage of the network resources that are being used by the waiting packet. The time of packet delivery thus depends on other packets flowing through the network. To ensure deterministic data delivery, all the traffic flowing through the network must be closely controlled.

A related problem occurs when a destination is not ready to receive and process a SpaceWire packet. If the destination node is not ready the packet will be strung out across the network blocking the links that it is resting on and temporarily preventing them from being used to transfer any other packets. So, not only is there a need to control packets going onto the network, but there is a need to ensure that once a packet arrives at its destination it is dealt with and removed from the network quickly.

Many components and sub-systems have already been designed using SpaceWire and ideally any deterministic data delivery mechanism for SpaceWire should be compatible with existing SpaceWire components. In any case it ought to be fully compatible with the SpaceWire standard.

3 SCHEDULING SPACEWIRE TRAFFIC

To prevent conflicting use of network resources the traffic has to be scheduled to avoid those possible conflicts. This requires a schedule table that defines which nodes can send packets at a particular time and some means of synchronising the nodes on the network so they can all follow the schedule. SpaceWire time-codes provide a mechanism for broadcasting time or synchronisation information across a SpaceWire network. The schedule is constructed from a series of periodic time-slots. The start of each time-slot is indicated by the arrival of a time-code so that all nodes are kept in synchronisation (see Figure 1).

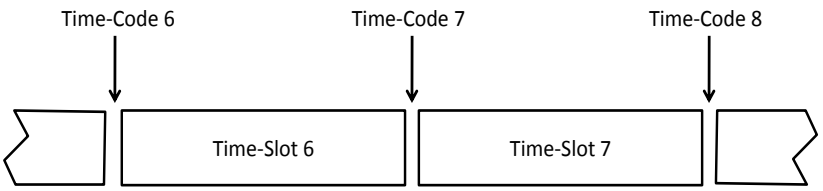


Figure 1 Time-Slots

4 WHAT SHOULD BE SCHEDULED?

Consider a system that schedules the sending of SpaceWire packets. A command may be sent in a SpaceWire packet to another node requesting that it returns some data. There is an asymmetry in the size of the packet used to send the command which is short and the reply containing the data which can be relatively long. Time-slots of equal period would not be very efficient in this case: a short time-slot is required followed by a longer one. If the command is sending data to the destination node and an acknowledgement is required then a long time-slot is required followed by a shorter one.

To mitigate the problem the complete transaction could be scheduled instead of the sending of individual SpaceWire packets. A transaction would then cover both sending the command and receiving the reply. The Remote Memory Access Protocol (RMAP) provides a transaction layer protocol which is able to read or write to memory in a remote SpaceWire node [3]. Designed for configuring, controlling and collecting data from instruments and other sub-systems it is being used in SpaceWire based systems. Components supporting RMAP include the ESA/UoD SpW-10X router [4] and ESA/Saab Remote Terminal Controller (RTC) device [5]. Missions using RMAP include Bepicolombo, ExoMars and MMS [6].

5 SPACEWIRE-D

To make maximum use of existing SpaceWire technology and devices, RMAP is used as the basic communication mechanism for SpaceWire-D and time-codes are used as the synchronisation mechanism.

Operation of the SpaceWire-D network is governed by a schedule that defines which node is allowed to initiate an RMAP transaction at any particular time. An example simple schedule table is illustrated in Figure 2.

Time-slot	0	1	2	3	...	63
Initiator allowed to initiate an RMAP transaction (logical address).	41	56	43	41		98

Figure 2 Example Simple Schedule Table

Each time-slot is long enough to allow one complete RMAP transaction to take place, allowing an RMAP initiator to read or write information to a remote RMAP target device. Since RMAP can transfer large amounts of data in a single transaction, which would take a long time, it is necessary to restrict the maximum amount of data that can be transferred in a single RMAP operation, so that the RMAP transaction does not take longer than one time-slot (but see section 6.3 later).

Each RMAP initiator has a copy of the schedule table. When a valid time-code arrives signalling the start of the next time-slot, each initiator checks whether it is allowed to start an RMAP transaction in that time-slot. If it has permission the initiator will send an RMAP command and wait for the RMAP reply. The RMAP command will travel across the SpaceWire network to the target node without any delay caused by network blockage, assuming that the schedule has been specified correctly.

When the RMAP command arrives at the target device, it is processed according to the RMAP standard and data read from or written to memory in the target device. It is important that this happens fairly quickly without the RMAP command being held up on the network. The RMAP reply containing data for a read command or an acknowledgement for a write command is then sent back across the network to the initiator.

Once the RMAP transaction is complete the initiator waits for the start of the next time-slot and then determines if it is allowed to send another RMAP command in the next time-slot.

6 SCHEDULE TYPES

There are three types of schedule defined in the SpaceWire-D specification: simple, concurrent and multi-slot schedules.

6.1 SIMPLE SCHEDULE

The simple schedule has been illustrated in Figure 2. It gives one specific initiator full control of the network for one or more specified time-slots. This means that when that initiator is permitted to send an RMAP transaction it may do so to ANY target node on the network. The RMAP transaction must start and finish in the same time-slot.

6.2 CONCURRENT SCHEDULE

The concurrent schedule makes more efficient use of network bandwidth by allowing more than one initiator to initiate RMAP transactions in a time-slot. Additional network performance is gained at the expense of a more complex schedule. This gives rise to the possibility that two initiators might attempt to use the same network resources (SpaceWire links) at the same time. The schedule table has to be constructed to prevent this. More than one initiator may initiate RMAP transactions in the same time-slot provided that the paths from each of the initiators to their targets do not use any of the same SpaceWire links in the network.

A typical application is on board data handling, where a mass memory unit is reading data from each instrument and writing data to a telemetry system, while a control processor is controlling instruments and monitoring housekeeping information. An example schedule table is illustrated in Figure 3.

Time-slot	0	1	2	3	...	63
Control Processor Targets	41, 43, 44, 45,	42, 43	42, 43	40, 41, 43, 44,		40
Mass Memory Targets	40	41	41	42		49

Figure 3 Example Concurrent Schedule Table

In this example the control processor can initiate RMAP transactions with one of several target devices listed in the schedule table for each time-slot. The mass memory device is gathering information from one target device in each time-slot. The targets that the control process can communicate with are carefully chosen to avoid any network resource conflicts with the transactions that the mass memory device is initiating in each time-slot.

6.3 MULTI-SLOT SCHEDULE

The multi-slot schedule builds on the concurrent schedule to improve network efficiency further. Where a large amount of data has to be transferred between two nodes, the RMAP transaction to accomplish this is permitted to occupy more than one adjacent time-slot. This allows more data to be transferred in the one RMAP transaction. The schedule has to ensure that no conflict of network resources occurs over the duration of this extended RMAP transaction. Additional network performance is one again achieved at the expense of a more complex schedule.

The schedule table has to ensure that when an RMAP transaction duration has more than one time-slot, it does not use the same network resources (SpaceWire links) as any other transactions occurring during any of those time-slots. An example schedule table is illustrated in Figure 4.

Time-slot	0	1	2	3	...	63
Control Processor Targets	41	42	43			40
Mass Memory Targets	40	41		42		49

Figure 4 Example Multi-Slot Schedule Table

In this example the mass memory initiates a long RMAP transaction with target 41 in time-slot 1. This transaction is expected to complete within two time-slots.

7 INITIATOR AND TARGET CONSTRAINTS

The time for the complete RMAP transaction to take place must be within the limits of the time-slot, or one transaction will have an impact on subsequent transactions. To achieve this, certain constraints are placed on the initiator and the target nodes. SpaceWire-D initiator and target implementations are fully compatible with the SpaceWire and RMAP standard. The constraints listed place some restrictions on the implementation covering the amount of data that can be transferred in a single RMAP transaction and the speed of response of the initiator and target devices.

7.1 INITIATOR CONSTRAINTS

The following constraints apply to the RMAP Initiator:

- The maximum amount of data that can be read in an RMAP read command or written in an RMAP write command is 256 bytes (TBC). This limits the size of the RMAP write command or RMAP read reply so that it does not exceed the duration of the time-slot. There is a trade-off between amount of data transferred in a single RMAP transaction and performance. Sending more data is more efficient giving an improved overall data rate, but the time-slot period then has to be longer making the delivery of data less timely.

- The maximum amount of data may be longer than 256 bytes when multi-slot scheduling is being used.

- The time taken from the receipt of a time-code to starting to send out an RMAP command from an initiator must be less than 5 μs (TBC). Note if this is difficult to achieve with a specific implementation of an initiator, operating

a local clock synchronised to time-codes might help with achieving this requirement.

7.2 TARGET CONSTRAINTS

The following constraints apply to the RMAP targets:

- The time taken from receipt of the complete RMAP command header in a target node to the authorisation or rejection of that RMAP command must be less than 5 μs (TBC).
- The latency in transferring data from SpaceWire interface to memory must be less than 5 μs (TBC).
- The time taken from completion of writing data to memory to starting to send the RMAP command must be less than 5 μs (TBC).

These constraints are readily met by the SpW-10X and RTC devices.

8 PERFORMANCE

In this section the anticipated performance of SpaceWire-D is considered. The operation of an initiator and target performing a write operation during a time-slot is illustrated in Figure 5.

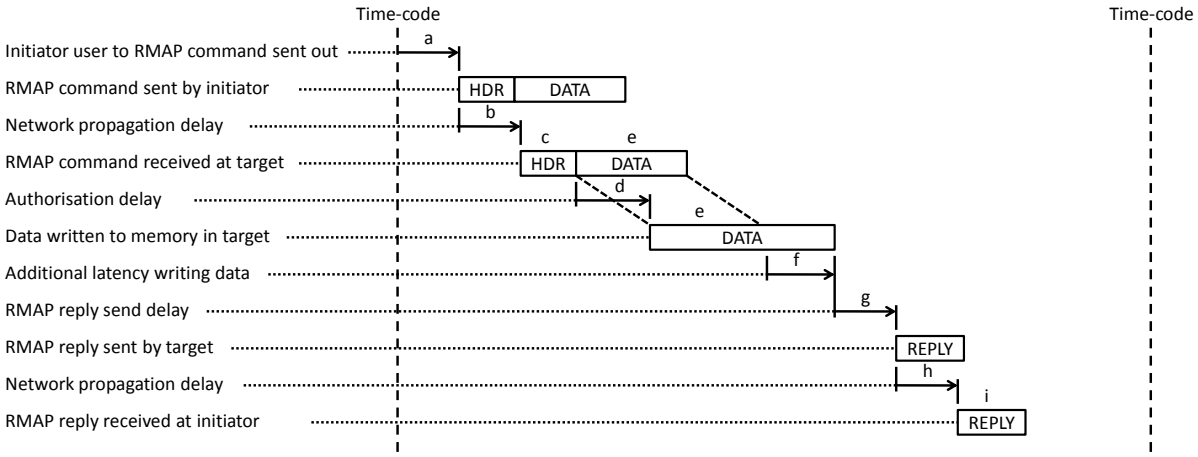


Figure 5 Performance of RMAP Write

The receipt of a time-code starts off the activities indicated in Figure 5. The following paragraphs explain each time interval labelled in Figure 5.

a) Interval from receipt of time-code to the RMAP command starting to be sent by the initiator. This interval includes: the time to receive, decode and respond to the time-code; the time to check the schedule table; the time to start to send out the RMAP command (assuming that the command has already been prepared ready for sending). This interval is entirely dependent upon the initiator implementation.

b) Interval for the SpaceWire packet containing the RMAP command to propagate across the SpaceWire network from initiator to target. This will mainly depend upon the number of routers between the initiator and the furthest target node. Assuming a time delay per router of 0.6 μs, the total propagation delay will be 0.6R μs where R is the number of routers in the longest path used between an initiator and a target.

c) Interval for sending the RMAP header, including any path address bytes. The size of the RMAP header including the SpW Target Address and Reply Address is H=R+16+P bytes, where R is the number of routers in the path from initiator to target and P is a the closest multiple of 4 which is greater than or equal to R. Thus if there are four router R=4 and P=4, so H=24 bytes. The time to send this header depends upon the SpaceWire data rate, S Mbits/s, and is 10H/S μs. For example with H=24 and S = 200 Mbits/s, T_c = 1.2 μs.

d) Interval for authorising the RMAP command once the header has been received. This is dependent upon the target implementation.

e) Interval to send the data and data CRC. This is given by 10(D+1)/S, where D is the number of data bytes. For D = 256 (the maximum amount of data permitted in a SpW-D RMAP write command or read reply) the time to send the data and data CRC is T_e = 12.85 μs when S = 200 Mbits/s.

f) Assuming that the target is able to write data to memory as fast as the SpaceWire network can deliver it, this interval covers any additional latency in the transfer of data from the SpaceWire interface to memory. It is dependent upon the implementation of the target node.

g) Interval from the completion of writing data to memory in the target to starting to send the RMAP reply. This interval is dependent upon the implementation of the target node.

h) Interval for the SpaceWire packet containing the RMAP reply to propagate across the SpaceWire network from target to initiator. This will mainly depend upon the number of routers between the initiator and the furthest target node. Assuming a time delay per router of 0.6 μs, the total propagation delay will be 0.6R where R is the number of routers in the longest path used between an initiator and a target. This interval is the same as (b).

i) Interval for sending the RMAP reply, including any path address bytes. The size of the RMAP reply including the Reply Address is E=R+8 bytes, where R is the number of routers in the path from target to initiator. Thus if there are four router R=4, E=12 bytes. The time to send this header depends upon the SpaceWire data rate, S Mbits/s, and is 10E/S μs. For example with E=12 and S = 200 Mbits/s, T_i = 0.6 μs.

The total time for the complete transaction is:

$$T_{total} = T_a + T_b + T_c + T_d + T_e + T_f + T_g + T_h + T_i$$

The corresponding performance of a SpaceWire-D network using simple scheduling is illustrated in Figure 6. The RMAP read operation has similar performance. The following assumptions were made:

- The link data rate is 200 Mbits/s,
- A simple schedule is used with one initiator initiating transactions at any time,
- The traffic comprised both payload data transfers (256 bytes per RMAP transaction) and command and control information (4 bytes per RMAP transaction) with an average of 132 bytes per RMAP transaction.

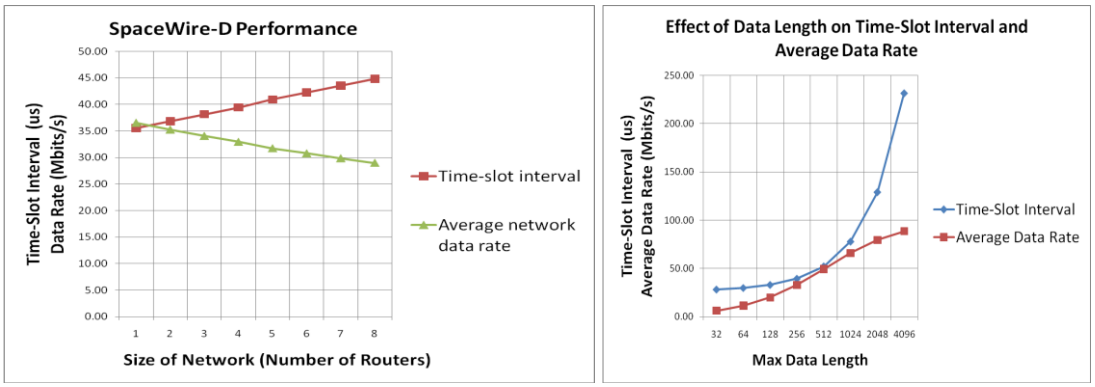


Figure 6 SpaceWire-D Performance

For a network with several layers of routing and links running at 200 Mbits/s, the overall data rate of the simple schedule system is around 30 Mbits/s (real data rate).

Initial SpaceWire-D prototyping done at Dundee to confirms this performance [7].

The performance of a system using concurrent scheduling depends on the number of concurrent initiators and the extent to which they can be scheduled to avoid use of common SpaceWire links. The maximum performance is N times that of the simple schedule, where N is the number of concurrent initiators.

The multi-slot schedule can substantially improve performance when there are large amounts of data to be transferred.

9 DEMONSTRATION SYSTEM

An implementation of SpaceWire-D has been developed by STAR-Dundee and the University of Dundee. This demonstration system used STAR-Dundee SpaceWire interface and routing devices. It also tested operation with an RTC device [7]. A screen shot of the demonstration system is shown in Figure 7.

Figure 7 Screen Shot from SpaceWire-D Demonstration System

10 CONCLUSIONS

SpaceWire-D provides a means of providing deterministic data delivery over SpaceWire. It builds on RMAP and is fully compliant to the existing SpaceWire standard. Furthermore existing components can be used in many cases without modification provided they meet some straightforward timing constraints. A demonstration system has been developed which showed SpaceWire-D operating as expected with multiple initiators and different types of target device.

11 REFERENCES

1. ECSS, "SpaceWire - Links, nodes, routers and networks", ECSS-E-ST-50-12C, July 2008, available from http://www.ecss.nl.

2. S. Parkes and A. Ferrer-Florit, "SpaceWire-D Deterministic Control and Data Delivery Over SpaceWire Networks", ESA Contract No. 220774-07-NL/LvH, University of Dundee, April 2010, available from http://spacewire.esa.int/WG/SpaceWire/.

3. ECSS, "SpaceWire - Remote memory access protocol" ECSS-E-ST-50-52C, 5 February 2010, available from http://www.ecss.nl.

4. Atmel, "AT7910E SpW-10X SpaceWire Router", available from http://www.atmel.com/dyn/products/devices.asp?family_id=641.

5. ESA/Saab, "SpaceWire Remote Terminal Controller", http://spacewire.esa.int/content/Devices/RTC.php.

6. D. Roberts and S. Parkes, "SpaceWire Missions and Applications", International SpaceWire Conference, St Petersburg, Russia, June 2010.

7. A. Ferrer and S. Parkes, "SpaceWire-D Prototyping", International SpaceWire Conference, St Petersburg, Russia, June 2010.

SPACEFIBRE

Session: SpaceWire Standardisation

Short Paper

Steve Parkes, Chris McClements

Space Technology Centre, School of Computing, University of Dundee,

Dundee, Scotland, UK

Martin Suess

ESTEC, ESA, Noordwijk, The Netherlands,

E-mail: sparkes@computing.dundee.ac.uk, cmcclements@computing.dundee.ac.uk, martin.suess@esa.int

ABSTRACT

SpaceFibre [1] is a very high-speed serial communications link being designed for use on spacecraft. It is designed to operate at speeds of 2 Gbits/s or higher depending on the specific driver/receiver technology used. Copper or fibre optic physical layers can be used. SpaceFibre is designed to interoperate with a SpaceWire network with a single SpaceFibre link being able to carry data from many SpaceWire links.

This paper provides a brief introduction to SpaceFibre and outlines the results of several SpaceFibre prototypes. It then considers how quality of service (QoS) will be implemented in SpaceFibre. The application of SpaceFibre in instruments, mass memory and processing systems is then described. The paper concludes with an overview of the current state of the SpaceFibre specification.

1 INTRODUCTION

SpaceWire [2] provides point-to-point and networked payload communication services for use on board spacecraft. It connects instruments to mass memory units and processing systems and provides the connection from the mass memory to the downlink telemetry system. SpaceWire uses bi-directional data links that operate up to 200 Mbits/s using current radiation tolerant components and micro-miniature D-type connectors. Higher speed operation is possible when matched impedance connectors are used. SpaceWire is being used on many space missions across the world. This success is due to many factors including standardisation, simplicity of implementation, performance and flexibility.

Several instruments, including synthetic aperture radar and multi-spectral imagers, require higher data rates to the mass memory unit. Downlink telemetry systems are being designed that can support Gbit/s data transfer leading to the need for similar data rates to transfer the data from the mass memory unit. There is a growing requirement for a data communication link with an order of magnitude higher performance than SpaceWire. Standardisation, simplicity of implementation and

flexibility are also import characteristics that need to be provided for a new data link technology to be successful.

The University of Dundee have been working on a Gbit/s data link technology for several years [3]. Trade-offs of ground data link technologies that could possibly be used as the basis for a new spacecraft Gbit/s data link have been carried out. An outline specification for SpaceFibre has been written and prototype implementations have been implemented and tested. Extensive work has been carried out for ESA on the physical layer and fibre optic components by Patria Oy, VTT, Fibre Pulse, INO and Gore [4][5]. Components have been selected and tested for flight applications.

2 SPACEFIBRE CODEC

The SpaceFibre CODEC [3] is responsible for the encoding and decoding of the data being sent over the communications link. A block diagram of the SpaceFibre CODEC is illustrated in Figure 1.

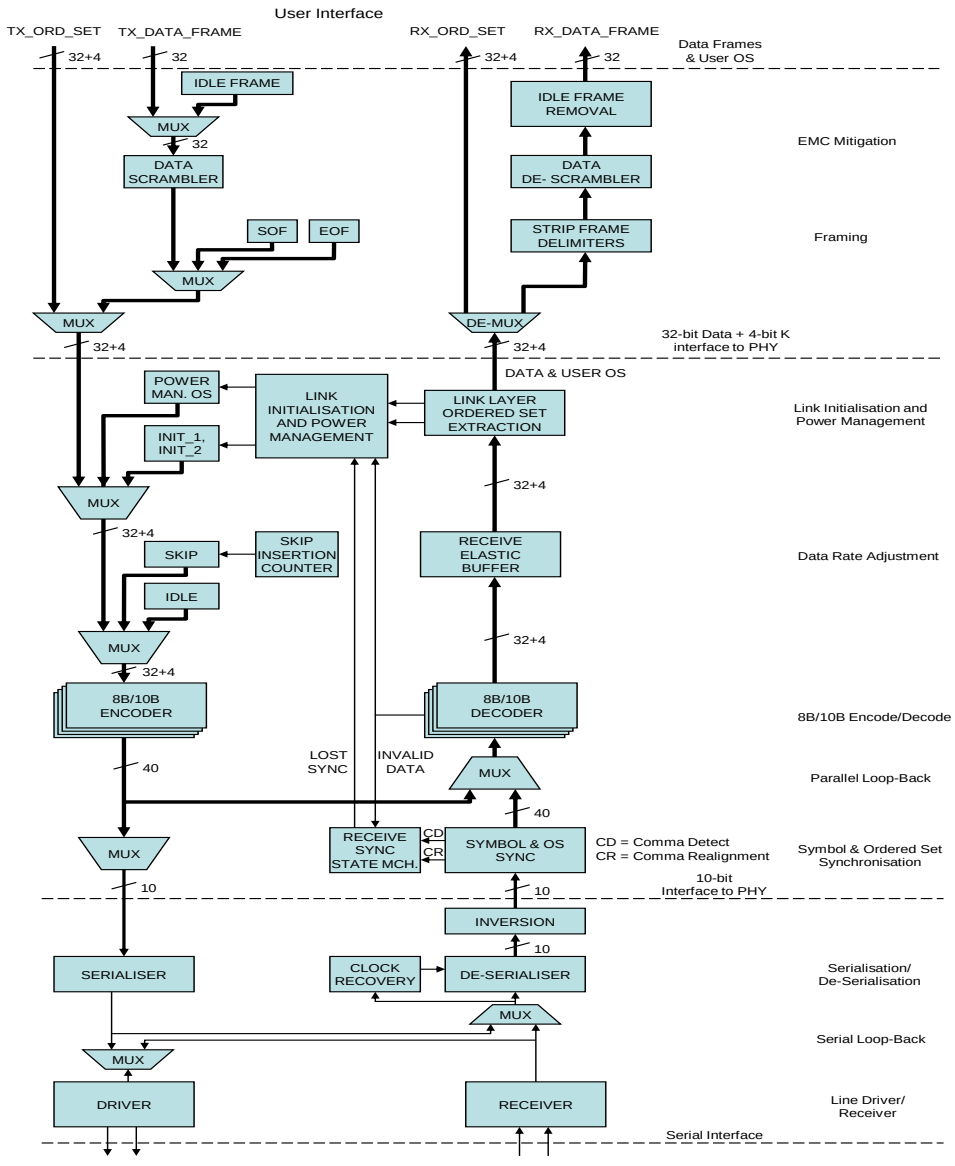


Figure 1: SpaceFibre CODEC Block Diagram

3 PROTOTYPE IMPLEMENTATIONS

Several SpaceFibre CODEC prototypes have been implemented by various organisations.

3.1 ESA/UNIVERSITY OF DUNDEE

University of Dundee have implemented several SpaceFibre prototypes. One of these, built for ESA, was integrated with the Fibre optic components provided by Patria and successful tests made at 2 Gbits/s over 100 m fibre optic cable.

3.2 NASA GSFC

NASA Goddard Space Flight Center (GSFC) implemented a SpaceFibre prototype and demonstrated it on the Max Launch Abort System (MLAS) [6].

3.3 JAXA

JAXA have also implemented a SpaceFibre prototype. This uses the "Wizard link" technology from Texas Instruments which provides 8B/10B encoding/decoding and serialisation/de-serialisation.

3.4 INTEROPERABILITY TESTING

Interoperability testing between the JAXA prototype and the University of Dundee prototype revealed some problems which are believed to be related to the use of the Wizard link device. The SpaceFibre specification will be altered to permit the use of the Wizard link as the lower level of the CODEC, as this is technology is available in a radiation tolerant device. Further interoperability testing is planned for later in 2010.

4 SPACEFIBRE QUALITY OF SERVICE

The SpaceFibre CODEC provides the basic means of transferring data over a SpaceFibre link, with data frames being used as a means of multiplexing several different data streams over the one physical link. To support several data streams running over the one link, virtual channels are provided as illustrated in Figure 2.

Data to sent over the SpaceFibre link is written into one of the virtual channel transmit buffers (VC TX Buffer). There is a separate virtual channel for each "stream" of data being sent over the network. The "stream" may contain individual commands, packets of data, or continuous data streams. The local system may use several virtual channels to send different types of data over the SpaceFibre network. There may be data waiting to be sent in several of the VC TX Buffers. The medium access controller (MAC) determines which VC TX Buffer will be allowed to send data when the current data frame has finished being sent. To be allowed to send data there must be data in the VC TX Buffer to be sent and there must be room in the corresponding VC TX buffer at the other end of the SpaceFibre link. Flow control information is passed from the VC RX Buffers at one end of the link to the MAC at the other end of the link so that the MAC knows which VC RX Buffer can accept data. If there is more than one VC TX Buffer with data to send with room in the corresponding VC RX Buffer, then a medium access policy will determine which of the VC TX Buffers is allowed to send data. SpaceFibre will support several QoS classes which drive the medium

access policy. These QoS classes include priority, bandwidth reservation and scheduled data transfer. A data retry mechanism is provided for each virtual channel which can be used to support reliable data transfer.

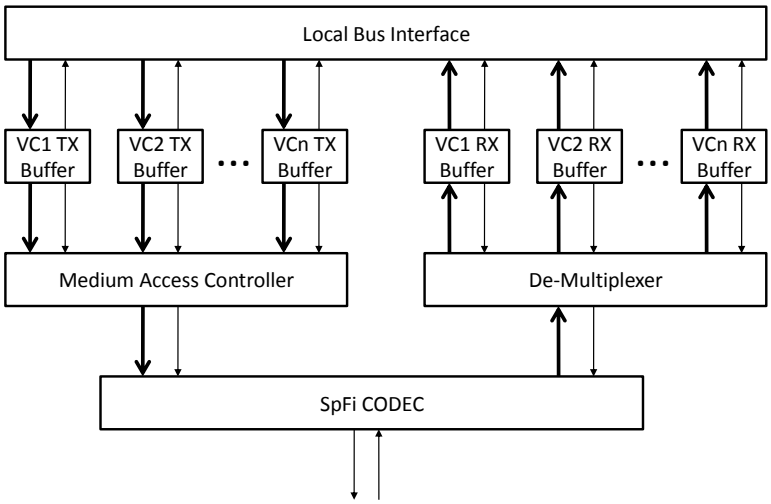


Figure 2: SpaceFibre Virtual Channels

When a VC TX Buffer is given permission to send data it passes a data frame of up to 256 bytes to the SpaceFibre CODEC for sending across the SpaceFibre link. The data is received at the other end of the link and passed via a data frame de-multiplexer to the appropriate VC RX Buffer. Data in the VC RX Buffers can be read by the local system.

5 SPACEFIBRE APPLICATIONS

SpaceFibre is being developed to support a wide range of spacecraft applications where high data rates are required. Specific applications include the interconnection of high data rate instruments to mass memory units, the transfer of data from the mass memory unit to the downlink telemetry system, and the multiplexing of data from several SpaceWire links over a single SpaceFibre backbone to reduce mass and power consumption.

6 SPACEFIBRE CURRENT STATE

The lower levels of SpaceFibre have been specified by the University of Dundee and a prototype SpaceFibre interface developed and tested. NASA have developed a prototype SpaceFibre system to this specification and tested it on the MAX Launch Abort System (MLAS) test vehicle. JAXA have also developed a prototype SpaceFibre system. QoS mechanisms for SpaceFibre are currently being defined and prototyped to provide a comprehensive capability set for future space missions. SpaceFibre is now being defined for use in several onboard applications including mass memory devices and DSP processors. SpaceFibre fills a growing gap in onboard communications links for spacecraft, which is being widened by the high data-rate demands of new instruments.

7 REFERENCES

1. S.M. Parkes. C. McClements and M. Dunstan, "SpaceFibre Outline Specification", University of Dundee, 31st Oct 2007.

2. ECSS Standard ECSS-E-50-12A, "SpaceWire, Links, Nodes, Routers and Networks", Issue 1, European Cooperation for Space Data Standardization, February 2003.

3. S.M. Parkes. C. McClements, M. Dunstan and M. Suess, "SpaceFibre: Gbit/s Links For Use On board Spacecraft", International Astronautical Congress, Daejeon, Korea, 2009, paper IAC-09-B2.5.8.

4. V. Heikkinen et.al., "Fiber-Optic Transceiver Module for High-Speed Intrasatellite Networks", Journal of Lightwave Technology, Volume: 25 Issue: 5, May 2007, page(s): 1213–1223.

5. J. Toivonen, "SPACEFIBRE - High Speed Fibre Optic Links for Future Space Flight Missions", Sixth International Conference on Space Optics, Proceedings of ESA/CNES ICSO 2006, held 27–30 June 2006 at ESTEC, Noordwijk, The Netherlands.

6. M. L. Davis and G. P. Rakow, "MLAS SpaceFibre High-Speed Serial Gigabit Data Link Technology Demonstration", NESC Request No.: 07-060-I Dec 17, 2009, available from http://www.nasa.gov/offices/nesc/reports/spacefibre_tech_demo_prt.htm .

Test and Verification 1

NEW APPROACH AND TECHNIQUES FOR TESTING AND DIAGNOSIS OF SPACEWIRE NETWORKS

Session: SpaceWire Test and Verification

Short Paper

Stéphane Davy, Jacky Rozmus, Matthieu Salanave,

SKYLAB Industries, 42 avenue du Général de Croutte, 31100 Toulouse, France

Frédéric Pinsard

DSM/IRFU, Bât 141, CEA Saclay, 91191 Gif sur Yvette Cedex, France

Mansour Talhaoui

Aerospace Services International Company, 3 rue du romarin, Taieb M'hiri, Tunis, Tunisia

E-mail: spacewire@skylab-corporate.com, pinsard@cea.fr, talhaouimansour@gmail.com

ABSTRACT

In this paper, we present a new testing approach and techniques for compliant SpaceWire[1] devices and networks. This approach is based on the emulation principle by using both physical and virtual devices. The conducted work has been done with the collaboration of **CEA** and **Aerospace Services International Company**.
Most of today's remaining issues and bugs in relation to test and integration are usually due to incompatible hardware, or simply, software misunderstandings between distant engineering teams, a common situation in space projects. The idea here is to introduce a new concept for emulating SpW networks based on physical equipments and/or virtual nodes and routers. Each of them could be located either in the same laboratory or at different locations around the globe. As a result, troubleshooting is expected to be more efficient and can be performed at early stages of the development, therefore ensuring successful flight modules.

After introducing the physical and virtual elements features which will act as SpW nodes for the network, the paper will detail the complete traffiController^{4SpW} software suite and how its architecture and characteristics can speed up network dimensioning, software testing and validation in a given heterogeneous SpaceWire topology. The software can actually be very flexible, allowing substantial freedom of use, not just for potential *4SpW* products users but also for SpW users without hardware equipment and theoretically for any type of SpW test equipment.

NETWORK PHYSICAL DEVICES

One main element of the test bench is the **PCI^{4SpW}** product: a PCI board with four SpW nodes, using CEA IP core. In addition to being able to transfer data with flexible configuration for each port, it provides standard and high-resolution time-codes capability. A SpW conformance testing feature is also present in the FPGA System on Chip, providing means to test robustness of any IP core. The PCI driver which is provided with the board can handle different data formats, depending on the project's performance and software constraints. Two main data transfer formats are available:

- A *short* format: one 9bit SpW characters is stored over a single 16bit word.
- ***SpaceWire Interpreted Protocol* format** (SIP):
  - *Data_8* format: limited to 256 values: one byte per SpW character.
  - *Data_32* format limited to FIFO size of hardware: 4 SpW characters per 32bit word.

Regarding memory capacity, the PCI board features a local and configurable 32K x 36bit SRAM memory dedicated to incoming data while a dedicated non-configurable 28KB of FPGA block-memory can be used for outgoing data. A PCI[4SpW] emulator has been implemented in order to anticipate the board delivery during SKYLAB early software prototyping, and can now act as a physical device emulator.
The other key element of the test bench is the **smartCable[4SpW]** product: a USB High-Speed single SpW node miniaturized device, overmolded with a male microD9 connector. It can act as node-to-node analyzer as well, using a dedicated plug (*spy mode*). It provides oscilloscope and logical analyzer capability, with possible buffering into the main 512Mbit DDR SDRAM. In addition, a custom LVDS eye diagram feature is available in the electronic device. The smartCable driver can handle enough bandwidth for most of the projects' constraints. During the smartCable prototyping validation, a throughput of 32MBytes per second uni-directionally was measured. SmartCable driver also provides implementation of hardware SIP format conversion for better processing efficiency: dedicated RTL modules actually ensures SIP to SpW decoding and SpW to SIP encoding in the smartCable System on Chip. Using dedicated routines of the API, the smartCable device also provides:

- **eye d iagrams** for both incoming and outgoing Data+/- and Strobe+/- differential lines. The Analog to Digital converter used for processing the analog signals is a dual 11-bit ADC with a 900MHz bandwidth, for potential LVDS measurements up to the highest standard transfer speeds. A low noise multiplexing front-end ensures very low disturbances regarding SpaceWire lines. The eye diagram is accessible in the API through direct picture format and dedicated routine.
- a **digital oscilloscope** function, providing useful serial data debug information.
- an **analyzer** function (in *spy* mode only). The additional *plug* of the smartCable (a female to female-female assembly which can be plugged to the male connector) reconfigures the FPGA of the USB bridge into receiving-only lines, providing a non-intrusive analyzer for two potential communicating nodes of, for example, a given set of space equipments.

CompatibleCable[4SpW] products have been used with 10-meter Ethernet cables in order to interconnect PCI devices at a transmit clock speed of up to 200MHz. A third possible test equipment for the test bench, the **PCI Express[4SpW]** board, could not be used for this set of testing because it was still under validation at the time of writing this paper. However, data transfers from PCI Express to SpaceWire have been successfully tested on the prototype model by CEA/IRFU. The test results for this board are promising and the board is set to provide high-performance transfer, especially because of the theoretical multi Gbps throughput of the PLX device associated with the performances of a Virtex 4 device, all two consistent with more demanding *ground and space* applications. A PCI Express[4SpW] emulator has been developed in order to anticipate the board availability and can act as a physical device emulator as well, as previously described for the PCI board.These three physical devices can be managed by the traffiController[4SpW] software suite, in order to be controlled and used as SpW nodes for transmitting and receiving data. RMAP, as described in ECSS-E-ST-11C, can be associated to each of these physical nodes.

NETWORK VIRTUAL DEVICE

In addition to this set of hardware, traffiController[4SpW] is also capable of managing virtual devices. These consist of the following elements, which use files for dump and load of data:

- **virtual node:** a SpW object-oriented element providing communication features, not related to any particular hardware, but dealing with data to and from a text file. Virtual nodes can communicate with real or emulated nodes from real or emulated hardware. They can also be virtually connected to one virtual router port (see hereafter).
- **emulated smartCable, emulated PCI, emulated PCI Express devices:** emulated object-oriented elements providing features equivalent to the real hardware, with degraded performances, also dealing with input and output files. These emulators provide emulated nodes for the SpW network, with more faithful characteristics and API methods regarding SKYLAB *4SpW* products.
- **virtual router**, allowing SpW routing, whose implementation is inspired from the SpW 10x device, as described in its User Manual[2] and which supports Group Adaptative Routing. It can be configured with up to 31 ports and can be controlled via RMAP port number 0.
- **virtual logical analyzer**, which can be inserted between nodes of any kind (physical, emulated, virtual) or router ports, allowing data to be monitored or stored for debugging purposes. Four modes are available: step-by-step, blocking buffer, non-blocking buffer and continuous modes.
- **IP tunnels**, which can be used to connect multiple API locally or through an intranet/internet network. These IP tunnel use a peer to peer architecture.

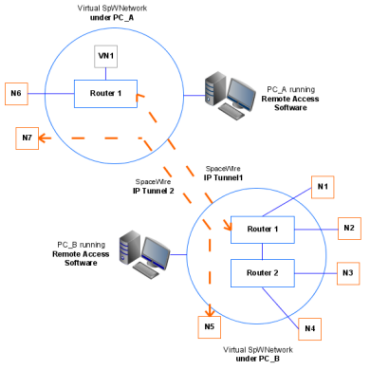


Figure 1: virtual SpW network and IP tunnelling

TRAFFICONTROLLER^{4SpW} ARCHITECTURE OVERVIEW

As previously described, the software suite enables space industry engineers to use network physical devices and network virtual devices. Emulated and real nodes are accessed through the Device Virtualization Service (DVS) layer, the lower layer of the traffiController^{4SpW} package. Virtual nodes are managed at API level, providing the rich set of communication elements for network routing and analysis, with the capability to transfer data through multiple tunnels. Additionally, Graphical User Interface (GUI) and console applications provide intuitive interaction with lower layer services. Each of these nodes (virtual, emulated and real) are accessed through DVS, whose architecture was inspired from the CCSDS SOIS PnP spatial related architecture[3]. More detailed information on DVS, API and GUI implementations is available online in the traffiController^{4SpW} product datasheet.

SPACEWIRE NETWORKS TESTING AND DIAGNOSIS: TEST CASES

Using previously described software and hardware resources, we could setup a reconfigurable emulated SpW network using multiple nodes potentially dispatched between Saclay (CEA), Toulouse (Skylab) and Tunis (ASIC) geographical sites. Such a test bench gave us flexibility to be able to configure sub-networks using virtual routers within the overall network. In the next test cases, we consider such a testing configuration based on a local API 0 configuration and a distant API 1 configuration, whose characteristics are described in Figures 3 and 4 below, with the related GUI screen copies. Note: 'T[]' stands for Tunnels in the related boxes.

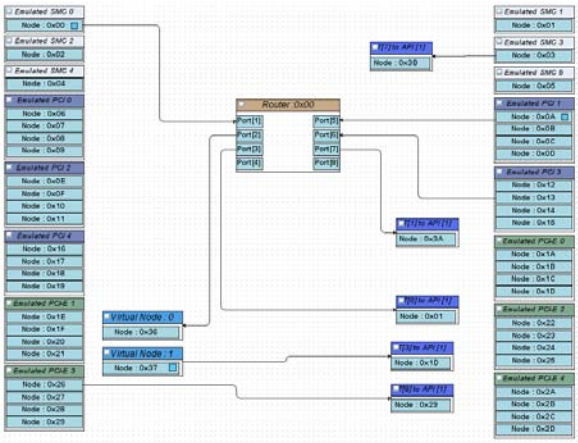


Figure 2 : API 0 configuration (local)

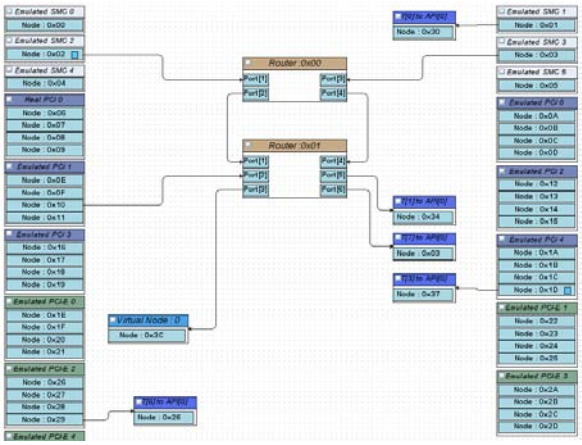


Figure 3 : API 1 configuration (distant)

These test bench configurations have been used to perform a few dozens of test scenarios which are summarized in the following three categories, according to their complexity:

1. Direct connection between nodes, with or without RMAP capability
2. Connection through just one router: many tests have been run using: (a) logical or physical addressing, (b) with or without RMAP, (c) router configuration packets
3. Connection through more than one router: same as above using intermediate routers

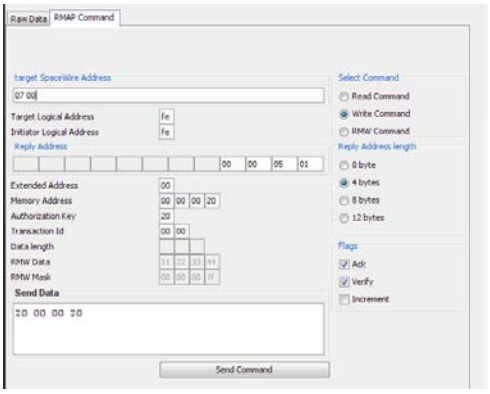
All these tests have been performed successfully using either a stand-alone configuration (API 0 and API running on the same PC) or a distributed configuration (API 0 and API running on different networks). For each of these tests, error codes processing have been treated to be able to handle bad commands, such as RMAP over SpW errors. The following are examples of test cases that were carried out:

- Test 1: Packet exchange between two remote nodes through a local and a remote router
- Test 2: Configuring a remote router
- Test 3: Using the analyzer function in *Blocking Buffer* mode

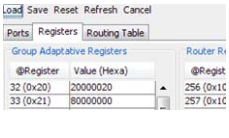
Test1: The objective was to enable the transmission and reception of data between two remote nodes through a local router and a remote router. The test case was to send data from the output file of Virtual Node 0 of API 0 while receiving them in the input file of the emulated smartCable of the API 1, located on anothercomputer.

Virtual Node 0 with ID 0x36 was selected to send various lengths and types of SpW data with the following specific physical addresses header: 0x07 0x01 0x03. When routed by Router 0x00, data were sent through the API tunnel T[1] with ID 0x3A, arriving Port 5 of Router 0x01 API 1, routed to port[1], routed to port[3] of Router 0x00 API 1 finally reaching emulated smartCable ID 3 of the API 1.

Test 2: The goal was to setup a router register using a Write Single Address RMAP command. The test case intended to configure Router 0x0 of API 0 from emulated smartCable 0.



As described on the left, a Write Single Address RMAP command could be sent from the *Test Node* window and checked that it was performed correctly by viewing the response in the Raw Data tab. The content of the register located at address 0x20 of the router could be checked as described in the following screen-copy.



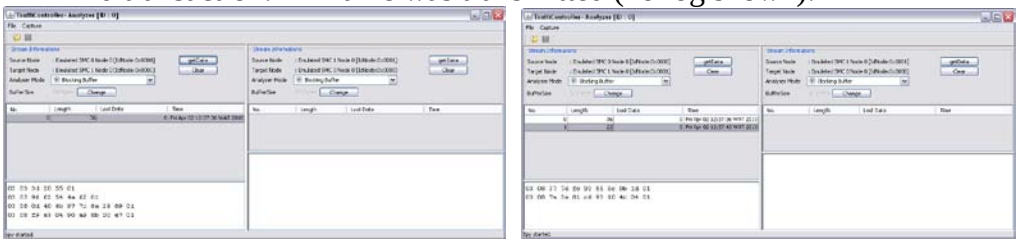
Test 3: The objective was to use the logical analyzer in *Blocking Buffer* mode, with a fixed buffer size of 50 bytes. The two following possibilities were considered: *sent data length* less than buffer size, then *sent data length* greater than buffer size: in this case, only the first frames had been recovered during the first reading. The normal behavior was that during this time, the next frame was on a pending state for the buffer release. All other sending attempts, after the buffer was full, had been ignored.

To illustrate the functioning of the *Blocking Buffer* mode, we considered theses hex frames:

Frame 1: 03 03 3d 20 55 01, 03 05 9f f2 54 4a f2 01
Frame 2: 03 08 0d 40 6b 97 7c 8a 29 89 01, 03 08 f9 a3 04 90 a9 8b 00 e7 01
Frame 3: 03 08 37 7d fc 93 55 5c 9b 1d 01, 03 08 7a 3a 81 c6 93 10 4c 04 01
Frame 4: 03 08 14 db 64 c6 d1 db 81 24 01, 03 06 a8 5c f7 53 a5 3d 01

The test was divided into three sequences:

- First transaction: 1st and 2nd frames were sent, as logged on the left window below.
- Second transaction: 3rd frame was then sent, as logged on the right window.
- Third transaction: 4th frame was transmitted (no log shown).



The first two frames were normally analyzed (sent data length is 36 less than 50 bytes). The second transaction was also analyzed because 3rd frame data was in pending mode. After completion of the first transaction, the available space in the buffer was only (50 – 36 =) 14 bytes and since the 3rd frame contained 22 bytes, it was immediately placed in a pending state to prevent data loss. The third transaction failed to be analyzed because the buffer was already full and there was another pending frame. This behavior complies with *Blocking Buffer* mode.

CONCLUSION

The paper introduced in details basic elements in test bench, typical API configurations and unitary/robustness tests and integration results of the test bench. The configuration aimed at representing most of the realistic scenarios met by node-to-node SpW users, router users and software developers. Most of these tests were conducted using representative emulators when drivers for physical devices were not yet available. Unfortunately, due to late driver validation for the smartCable and the unavailability of certain features like eye diagram or physical logical analyzer, unitary tests for these features could not be done and performance tests for this device with the traffiController could not be performed. However, in addition to the final PCI and smartCable performance tests, additional measurements will be done in the coming months with the new PCI Express equipment, giving complementary results to this approach.

REFERENCES

1 ECSS-E-ST-50-12C (Spacewire – Links, nodes, routers and networks)
2 UoD_SpW_10X_UserManual (SpW-10X SpaceWire Router - User Manual)
3 CCSDS-SOIS PnP on MILBUS (SOIS PnP Device and Service Discovery an example of an adaptation model for MILBUS (On ECSS-E-50-13 Compliant System))

Research of Application Simulation in Satellite Data Management System Based on SpaceWire

Session: Test and Verification
Short Paper

Wang Rui, Li Guoliang
China Aerospace Engineering Consultation Center
Cao Song
Center for Space Science and Applied Research, China Academy of Science
E-Mail: hustwr@yahoo.com.cn

Abstract

We use SMAT to design a model for the prototype of satellite data management system based on SpaceWire. This model is capable of dealing with situation where data rates vary in a large scope and space data systems in accordance with CCSDS international standard have multiple transaction structures. We use this model to make simulations of SpaceWire with the features of high speed transmission, network routing and redundance in linkage. In order to examine the feasibility of our experiments and the system performance, we make tests to provide system designers with more quantitative evidences.

1．Introduction

The tasks of Satellite data management system include program control of satellite, command delay, Payload Operation and Management, systems on satellite, and etc[1]. The AOS standard of Consultative Committee for Space Data Systems（CCSDS）integrates multiple data into a satellite data network, and it provides three services including data management, data routing and data transmission channel.

SpaceWire is being used successfully on many space exploration missions[2][3]. However, there are several problems SpaceWire must face in the design of a practical satellite data system, such as what kind of topological structure can improve efficiency and reliability of data exchange [4].

However, research of the problems above cannot rely on the actual network construction [5]. It is difficult to complete the research of performance and parameter test above in a relatively fixed physical network. While system simulation is a very good research means to resolve these problems.

2．Technique and Tools for Discrete Events Simulation

Based on the objective of system analysis, the system simulation establishes a simulation model which can describe the system structure or process of action and can be expressed by some logic or mathematical equation on the basis of analyzing the nature and relationship of the elements in the system. Accordingly, Experiments or quantitative analysis are made to obtain all the information required for proper

decision-making. By the simulation model, system simulation can successfully solve the system problems such as forecasting, analysis and evaluation for some object systems which are difficult to establish the physical and mathematical model.

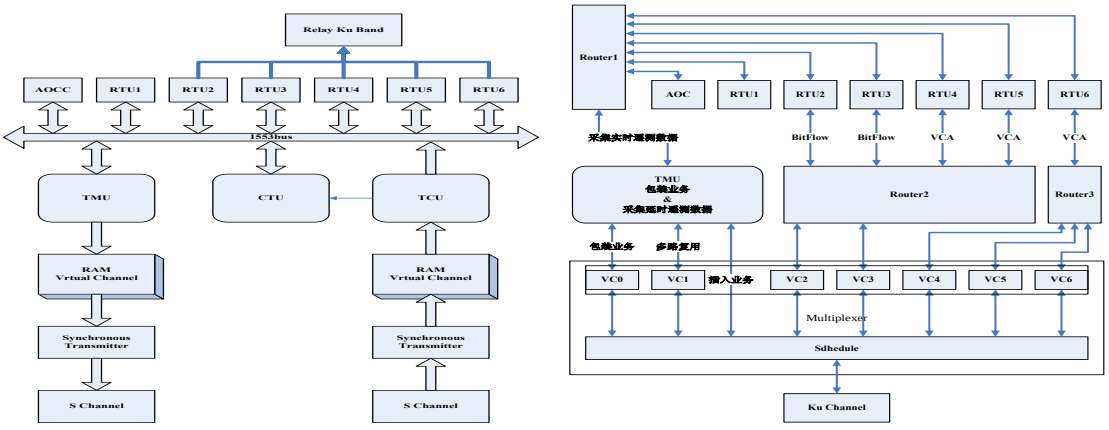
Based on the OMNET++, we developed the System Modeling & Analysis Tool. The tool can do the analysis, quantitative evaluation, validation and optimization for the system-level design. The whole process of system simulation is also supported by the tool.

3. System Modeling

Satellite Conventional Orbiting System (COS) is composed with five units including Central Control Unit (CTU), Telecommand Unit (TCU), Telemetry Unit (TMU), Attitude & Orbit Control Circuit (AOCC) and Remote Terminal Unit (RTU), all of which are connected with serial bus (Bus 1553 in practice).

CTU is in charge of transaction management, bus dispatch, etc of satellites. TMU is designed to collect real-time telemetering data from sub-systems and send them into downlink virtual channels and to the surface through channel S, while generating delayed telemetering packages according to specified sampling rates. TCU sends indirect remote instructions to devices respectively through uplink virtual channels, RTU is used to carry platforms and actual payloads from which large amount of data generated is sent to the surface through wave band “Ku” by specified devices. The diagram below demonstrates the structure of COS model.

Since CTU, TMU and TCU are different units taken from one single CPU, and the amount of uplink data is small, the uplink part with mono functions can be facilitated into remote instruction generators. The CTU and TCU are integrated into TMU.



With SpaceWire and FPGA, the bottleneck of application in AOS can be solved. The diagram above demonstrates how to use SMAT to build a SpaceWire and FPGA based AOS downlink model, in which all routers are SpaceWire based, Router1 is in full-duplex mode, Router2 and Router3 are in simplex mode, TMU is the real-time or delayed telemetering data collect unit, and Multiplexer is combiner.

3.1 Data Source Model

In satellite electronic devices, all payloads units, telemetering units and

telemetering units can be seen as data generators (data sources) obeying specified rules. AOS system is made up with eight transaction modules including routing module, internet module, package module, multiplex module, bit-stream module, virtual-channel-access module, virtual-channel-data-unit module and injection module. SMAT provides data source modules for all AOS transactions. As the structure of routing transaction is extremely complex and this transaction involves communication between multi-system and multi-satellites, it is rarely used in practice.

3.2 Router Model

Each node of the router contains four ports: Data input port “In”, Data output port “Out”, Node-state port “Status” for the situation when sending data from one node to another node and the port “Busy” to set the node refuses to accept data.

The module “Nod_In” receive data-input-request, if the output node being requested is servicing for the other input nodes, then return “Busy” flag, do not discard the data, enter the waiting service sequence，until the data is transferred then return “Finished” flag; If the corresponding FIFOBuffer of the output node has overflowed, then discard the data and return “Overflow” flag; Or else send the data to the corresponding module “Nod_Out”, set the flag to “Busy”, and then return “Finished” flag until receive the "No Busy" signal.

The module “Nod_Out” transmitted out the data it has received, and then send “No Busy” signal to the module “Nod_In”.

The Module "Error" produce interference signal according to the rules have been given, if detected signal “Noise” during data transfer process, then re-send the data and record the length of the re-send data simultaneously, to test fault tolerance ability of the router link-layer.

3.3 TMU Model

The Module “TMU ” collect telemetry data from each load, and then pack the data unit（E－SDU）, is arranged by byte and is not delimit by CCSDS structure ,into statute data unit E－PDU (CCSDS package) by fixed position . And then multiplexing with the other CCSDS packages to produce M-PDU package. And simultaneously extract data in the required Sampling rate, generate delayed telemetry data. For the data required real-time transmission, the “TMU” combined them into Insert data to generate IN-SDU.

In addition to the functions mentioned above, the module “TMU” also distributed telemetry command (Telecommand) to the corresponding load units. In this experiment, "TC Generator" replaced the actual uplink telecommand based on the ground control rules and produce regular telecommand.

3.4 Multiplexer Model

The Multiplexer schedule the VCDU data units of all virtual channels，add current IN-SDU header and VCDU header，in this way generate the VCDU. If there is none of the VCDU data units in the virtual channels, we should fill “0” in the VCDU data units and generate a leisure data frame. Coding the VCDU or leisure data frame,

we can finally generate the PCA_PDU to be sent.

4. Experiment Scheme

Models of the system are all achieved by SMAT simulation tool. And the models are placed in the model library. Different simulation systems can be constructed according to the purposes of experiments. In this paper, the simulation system is composed by the data transmitter and data receiver. Before running the simulation model there are the following work needed to carry out:

4.1 Verification of model accuracy

The verification of the correctness of the simulation model is to test the model constructed which can really represent the basic performance of a real system (or the system designed) or not. The correctness validation process for the simulation model is a process of repeated comparisons between the model and practical system. And the difference generated by comparison is used to improve and modify the model in order to make the model gradually approach to the actual system. The process will not stop until the simulation model is recognized as the true representative of the real system.

4.2 Confirmation of Data Payload

In our simulation experiment, the primary objective is to add data payload to the model. Since the amount of data sources is huge, we pick up several transaction data to be the object in channel combiner according to the features of AOS and the need of our research.

The table below shows the data sources employing AOS structure.

N.o	Data source	Service type	Affiliated	Data length (Bytes)	Speed (Mbit/s)	Distribution (s)	Virtual channel
1	Camera	B_PDU	RTU2	—	150		2
2	Env_Science		RTU3	—	50		2
3	Mat_Science	VCA_SDU	RTU4	—	30		3
4	Life_Science		RTU5	—	20		3
5	TV Video*3	B_PDU	RTU6	—	2.048		4，5，6
6	EngineeringTM	Package、multi duplicate	AOC	131934	-	4	0
7	Audio		RTU6	15360	-	2	0
8	Delay TM		ALL	Sampling	-		1
9	TM*7		ALL	40	-	0.512	0
10	Power		AOC	80	-	0.512	0
11	Self_manage		RTU1	10	-	0.512	0
12	Insert	IN_SDU		64	-	-	0

4.3 Simulation Model Performance Statistical Index Architecture

The final statistical indicators of performance extracting from our simulation experiment are composed with:

- Data transmitting availability of downlink channel
- Data transmitting immediacy
- Data capacity
- Data transmitting fault tolerance

4.4 Confirmation of Experiment Scheme

In this experiment, the model designed on the system's every function is followed AOS system architecture's function and target. The algorithm of the virtual channel schedule is a key points of the AOS channel combiner. A reasonable schedule algorithm is a assurance that AOS channel combiner will finish each complex task in order. The efficiency of the algorithm will make great affect on the system. In the modeling course, we compared several schedule and finally take the schedule of priority and overtime, which take the longest delay time as the standard of the transfer in real time.

We analyze downlink channel data transmission availability, data delay, data load capacity and data effective transmission rate with interference, which are captured in our experiment. Then we compare our results with the simulation outputs taken from conversional satellite orbiting system, while we can see the performance is improved in many aspects. Furthermore, utilizing the SpaceWire routers and without high speed CPU (DSP), it is feasible to break through the restriction of data stream, realize the design plan of AOS system, and improve the fault tolerance in the link layer.

5. Summary

We in this paper suggest a method to implement AOS system with SpaceWire routers, and make simulation experiment using real data source with our designed system model. The experiment shows the bottleneck in AOS system (low throughput capacity and low CPU speed) can be eliminated with the deployment of SpaceWire routers. By adjusting the topology structure, we can get system performances with different topology and same data payload, and thus optimize the system design.

6. Reference

[1] Parkes S, Rosello J, SpaceWire-Links, nodes, routers and networks, DASIA 2001 -Data Systems in Aerospace, Proceedings of the Conference, 2001.

[2] R Amini, E Gill, G Gaydadjiev, The Challenges of Intra-Spacecraft Wireless Data Interfacing, 57th International Astronautical Congress.

[3] W.J. Dally and C. Seitz, "The Torus Routing Chip", Distributed Computing, vol. 1, no. 3, 1986.

[4] SM Parkes, P Armbruster, SpaceWire: a spacecraft onboard network for real-time communications, Real Time Conf., 2005.

[5] C. McClements, S.M. Parkes, and A. Leon, "The SpaceWire CODEC," International SpaceWire Seminar, ESTEC Noordwijk, The Netherlands, November 2003.

EVALUATING SPACEWIRE SYSTEMS

Session: SpaceWire Test and Verification

Short Paper

Dr Barry M Cook, C Paul H Walker

4Links Limited

E-mail: barry@4Links.co.uk, paul@4Links.co.uk

ABSTRACT

The 4Links family of SpaceWire test equipment has been built over many years, and this paper will consolidate and update what has been presented at past ISC conferences and at SpaceWire Working Group meetings. The family consists of interfaces, monitors, and instrumented simulators, and a synchronization interface to absolute time. Versions are available with or without extensive instrumentation for measuring time and performance, and with or without synchronization across a complete SpaceWire test set. The equipment has been used for a wide variety of applications and some of these user scenarios will be described.

1 INTRODUCTION: THE 4LINKS FAMILY OF SPACEWIRE TEST EQUIPMENT

The 4Links family of SpaceWire test equipment started with the PCI-1355 board in 1997, which was used by ESA and several of the companies with early interest in the technology that evolved to SpaceWire. PCI-1355 was built with Inmos C101 chips, and the SpaceWire-PCI followed in 2000 using the ESA-sponsored SMCS chips. The SpaceWire-PCI board was widely used, by ESA, by NASA, and by many companies.

1.1 BENEFITS OF INTERFACING VIA ETHERNET AND TCP/IP

4Links introduced the first Ethernet/TCP/IP-based SpaceWire test product in 2004, and has built a comprehensive family that is regularly acclaimed by its users for its quality and usefulness.

Ethernet and TCP/IP are provided with every main-stream operating system and the user has no need to install, and keep updated, any other driver software. Users are able to work remotely, with the benefit of running the tests from the office instead of the clean room. Remote working can even extend to other countries and continents, and 4Links equipment has been used in this way to help users to diagnose problems.

The Ethernet/TCP/IP based test equipment family consists of interfaces, monitors, instrumented simulators, and a synchronization interface to absolute time. The products are accurate, reliable, use a uniquely mature synchronous SpaceWire Codec, and include failsafe measures to protect flight equipment from faults.

2 INTERFACES FROM A COMPUTER TO SPACEWIRE

A range of products offer a choice from a simple interface with one SpaceWire port to a full diagnostic interface with up to eight ports, and a new Wireless interface.

The **EtherSpaceLink** provides a single active SpaceWire port. This can be in a single port unit, or can be software selectable from up to eight physical SpaceWire ports.

The EtherSpaceLink proved its worth immediately. A test with one of the agencies, just before silicon tape-out, exposed a lost byte under certain conditions. Another early user needed the PC to run Real-Time Linux, and problems with a USB interface caused delays. The EtherSpaceLink ran immediately and progress was resumed.

The EtherSpaceLink has recently been upgraded to generate a Time Code from an external pulse or, as originally, from its internal clock.

The **Diagnostic SpaceWire Interface** is able to drive up to eight SpaceWire ports simultaneously and, as its name implies, includes comprehensive diagnostics.

One device tested by the Diagnostic SpaceWire interface was ESA's SMCS-SpW, unfortunately too late to prevent errors in the silicon. One of the errors seen could result in deadlock from a lack of flow-control credit, another was the delivery of a packet to the user despite there being a parity error in the packet and another was occasional failure to detect disconnection timeout as defined in the SpaceWire standard [1].

The Diagnostic SpaceWire interface is widely used for many purposes, including simulating, testing, debugging and validating both hardware and software of SpaceWire devices, boards, and subsystems.

The use of Ethernet and TCP/IP allow 4Links equipment to be in the clean room while the engineers testing the satellite are outside the clean room. For planetary landers or rovers, the need for cleanliness is particularly acute. Having a **SpaceWire WiFi Interface** built into the satellite could avoid the need for any other test equipment to be in the clean room. Under an ESA project with SEA, 4Links built a concept demonstrator for such an interface, housing the SpaceWire interface within a COTS Access Point. This will be re-engineered as a commercial product.

3 MONITORING, ANALYZING, TIME-TAGGING AND RECORDING SPACEWIRE

Active interfaces, and the software supplied with them, permit extensive analysis of equipment under test. But when equipments are connected together, there is also a need for passive monitoring. Monitors include an Analyzer that gathers statistics of the traffic, and a Recorder that records the traffic.

The **Multi-link SpaceWire Analyzer** accumulates statistics of the SpaceWire characters flowing in each direction of the SpaceWire links. It is an "honest broker" between flight units, or between a flight unit and test software under development. It has quickly enabled the solution of problems that had been delaying projects.

The **Multi-link SpaceWire Recorder** passively records traffic in each direction on each link. All the recordings are time tagged, and multiple recorders can be

synchronized so that recordings on different computers and different discs all have consistent time tags. The recorder is used for debugging higher level system behaviour, and for archiving the results of extensive tests. Recordings can be of any size, limited only by the disc capacity.

4 INSTRUMENTED SIMULATORS

The **Flexible SpaceWire Router** can be used as a packet routing switch, but also as a static routing switch, a mux/demux (concentrator/deconcentrator), as multiple small switches or one large switch comprising several units. It is also instrumented to provide analysis of traffic through the switch.

The **SpaceWire RMAP Responder** simulates instruments and memories that respond to RMAP commands. The memory accessed by the RMAP commands on each port is managed by the user from a PC. Response is exceptionally low latency (single-digit micro-seconds). Other protocols such as CCSDS packets, interleaved between RMAP packets, are passed transparently to and from the PC.

The **Absolute Time Interface** synchronizes a test system to the IRIG time standard, and provides a source of low-jitter Time Codes. Synchronization to IRIG is accurate to within far less than one microsecond, and synchronization between 4Links test units is of the order of ten nanoseconds. With synchronization to IRIG, time tags include both the date (day of year) and an indication of the accuracy of the time.

5 CAPABILITIES AVAILABLE ACROSS THE PRODUCT FAMILY

Versions of the products are available with or without extensive instrumentation for measuring time and performance, and with or without synchronization across a complete SpaceWire test set. These capabilities include:

- Error Injection, for testing the correct response to such errors
- Error Reporting, included in most products, but not always needed
- Event/Error Waveforms, invaluable for hardware debug
- Packet Statistics, simple, accurate, information independent of user software
- Synchronized Outputs, necessary for testing arbiters in routing switches
- Time Tagging, necessary for any time-dependent activity
- Time Code generators, autonomous, accurate, low jitter
- Hardware synchronization between units, essential for real-time system analysis

6 USE SCENARIOS

4Links test equipment addresses a wide range of user scenarios including testing and validating new designs, measuring performance under varied operating conditions, simulating units controlled by an OBC, and archiving the results of a long series of tests. These scenarios exist both with individual test units for a single subsystem and with many synchronized units for a complete mission. Examples are testing a routing switch, testing real-time protocols, and re-use of equipment for different tests.

6.1 TESTING A ROUTING SWITCH

Routing switch tests need to include synchronized packets with controllable offsets arriving at the routing switch, testing with several patterns such as all ports accessing the same port, each port accessing a different port, and random addressing, data, and packet lengths.

Such tests can be performed by a set of synchronized Diagnostic SpaceWire Interfaces, all driven by a single computer or each by a separate computer. Test unit synchronization enables the headers of all the packets to be aligned within a few nanoseconds, and time tags are matched so that records of the tests made on different computers can all be correlated with respect to time.

6.2 REAL-TIME PROTOCOLS

Real-time protocols need an accurate source of Time Codes. Such Time Codes can be generated either by the Absolute Time Interface from IRIG, or by the EtherSpaceLink interface.

The actual times of the real-time protocol packets, and of real-time events generated by them, can be recorded by one or more Multi-link SpaceWire Recorders, synchronized to the Time Code source and to each other.

6.3 RE-USE AND SPARES

During a project, the needs for test can change from needing active interfaces for test to using monitors for integration. The family of 4Links test equipment is based on a small set of hardware platforms which are customized to a particular product function by an exchangeable memory card. A hardware platform can be used with a variety of different memory cards. This can be useful, for example, to turn an interface used for testing early in a project into a monitoring recorder when the flight equipment is being integrated. It can also be useful to have a spare hardware platform — with memory cards for all the functions used in a test bench — as a standby unit, greatly reducing the cost of spares.

7 CONCLUSIONS

A wide range of SpaceWire interfaces, monitors, and instrumented simulators has been described. Users have found these interfaces highly effective in exposing previously undetected design anomalies, of quickly diagnosing and fixing such anomalies, and hence overcoming obstacles to progress. The range includes functions that can be used at all the stages of development through to integration. The hardware platform can often be re-used to minimize cost over the mission life-cycle. The quality and effectiveness of the equipment encourages feedback from users on potential improvements. Many of the capabilities described have resulted from such feedback, and the authors acknowledge and thank those who have provided it.

8 REFERENCES

1. “ECSS-E-ST-50-12C 31 July 2008, SpaceWire - Links, nodes, routers and networks”, published by the ECSS Secretariat, ESA-ESTEC, Requirements & Standards Division, Noordwijk, The Netherlands

Test and Verification 2

SPACEWIRE LINK ANALYSER MK2: A NEW ANALYSIS DEVICE FOR SPACEWIRE SYSTEMS

Session: SpaceWire Test and Verification

Short Paper

Pete Scott, Chris McClements, Stuart Mills

STAR-Dundee,

c/o School of Computing, University of Dundee, Dundee, DD1 4HN, Scotland, UK

Steve Parkes

School of Computing, University of Dundee, Dundee, DD1 4HN, Scotland

E-mail: pete@star-dundee.com, chris@star-dundee.com, stuart@star-dundee.com, sparkes@computing.dundee.ac.uk

ABSTRACT

The process of testing, validation and verification of a SpaceWire network [1] benefits from the ability to analyse the traffic passing over a SpaceWire link and to monitor its status. This paper gives an overview of the capabilities of two STAR-Dundee products which perform this important task: the SpaceWire Monitor and SpaceWire Link Analyser. Both of these devices' functionality is combined into a new product; the Link Analyser Mk2, which features interfaces to both a Logic Analyser and a Personal Computer (PC), external triggering capabilities and a large memory for capturing decoded data. Software running on the PC is capable of interpreting data captured by the Link Analyser Mk2 into a user-defined protocol.

1 INTRODUCTION

A typical SpaceWire system may consist of many SpaceWire components from different sources connected together through a network of routers and links which are often duplicated for redundancy. The testing, validation and verification of simple point to point connections or complex networks call for a tool to analyse the data flowing at any given time across a SpaceWire link.

2 EXISTING SPACEWIRE LINK ANALYSIS TOOLS

2.1 SPACEWIRE MONITOR

The SpaceWire Monitor [2] used together with a logic analyser provides a means of analysing the traffic flowing through a SpaceWire link. Figure 1 illustrates how two SpaceWire cables are used to connect the SpaceWire Monitor across a link to be monitored. The Monitor decodes the characters and symbols passing bi-directionally across a link and provides direct indication of link status and traffic flow on a series of LEDs.

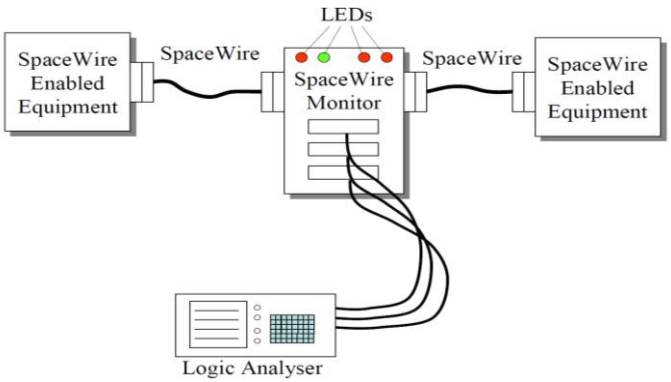


Figure 1: Using a SpaceWire Monitor and logic analyser to analyse a SpaceWire link.

For a deeper insight into the link under test, a logic analyser may be interfaced to the monitor to record the decoded values of characters and symbols as shown in Figure 2.

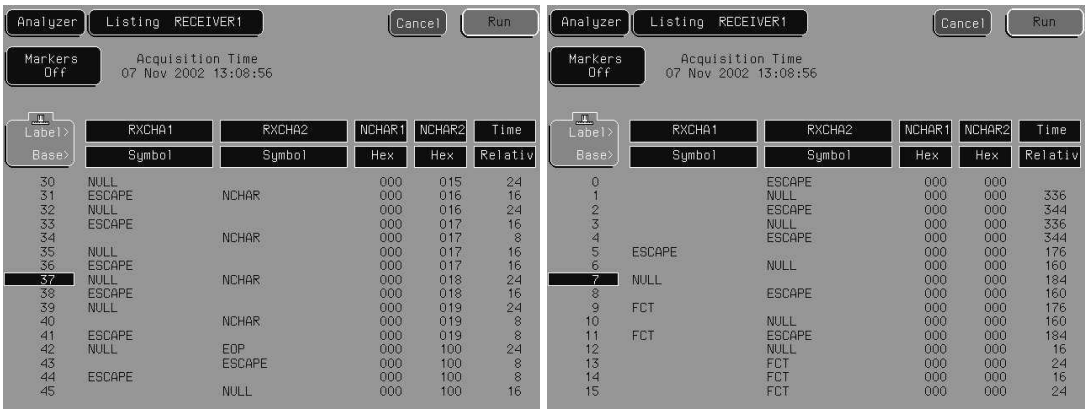


Figure 2: A SpaceWire ink during data transfer (left) and link start-up sequence (right).

The SpaceWire Monitor lacks external trigger signals to interface to the logic analyser. Triggers can only be programmed into the interfaced logic analyser and errors latched onto the relevant indicator LEDs.

2.2 SpaceWire Link Analyser

The STAR-Dundee SpaceWire Link Analyser [3] provides link monitoring and analysis capabilities with inbuilt logic analysis functionality. It is connected across a SpaceWire link in a similar fashion to the SpaceWire Monitor. A host PC is interfaced to the Link Analyser via a high speed USB 2.0 interface shown in Figure 3.

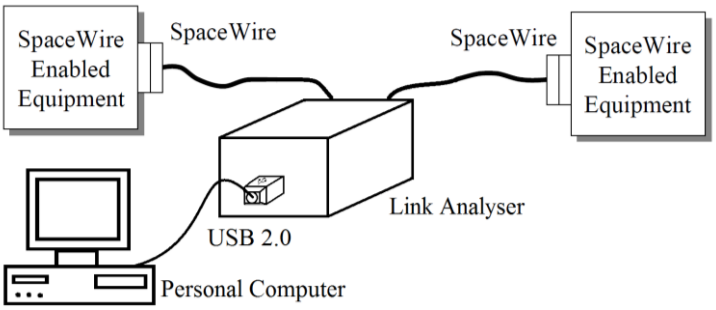


Figure 3: Using a SpaceWire Link Analyser and PC to analyse a SpaceWire link.

Data passing across a link under test are decoded into characters and symbols which are recorded into the Link Analyser memory. Software running on the PC is used to configure a sequence of triggers in the Link Analyser to activate on a change in state of the link status and/or a series of characters flowing over the link. Statistical information on the level of traffic through the link is continuously gathered and displayed on the host PC as shown in Figure 4.

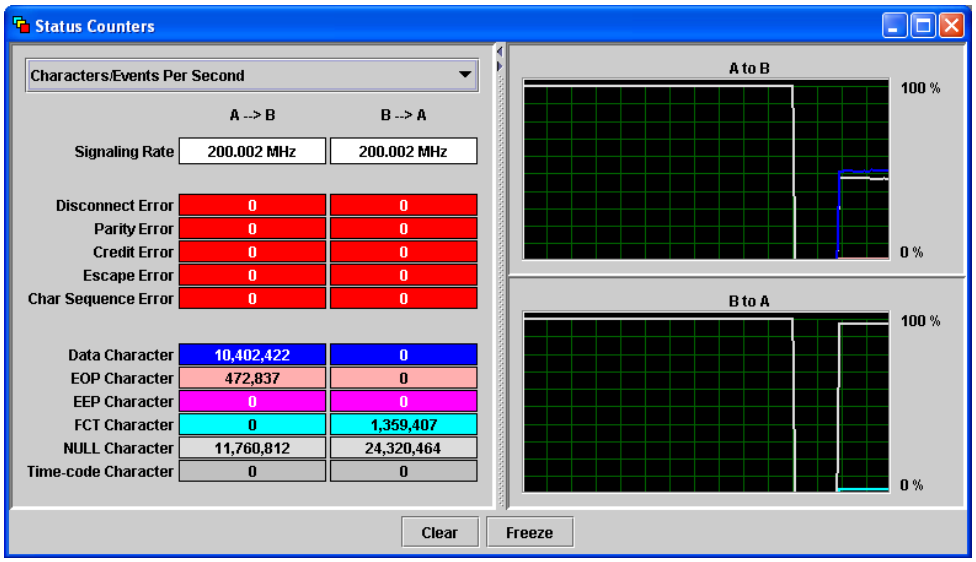


Figure 4: Display of link level activity and statistics of analysed link activity.

The Link Analyser is capable of monitoring, tracing and recording traffic at the link level to confirm link start-up, flow-control, data transfer and error recovery. Data can be monitored, traced and recorded at the packet level to confirm the response of a system to packet errors and the control of SpaceWire systems using control packets. The link and packet level data is shown in Figure 5.

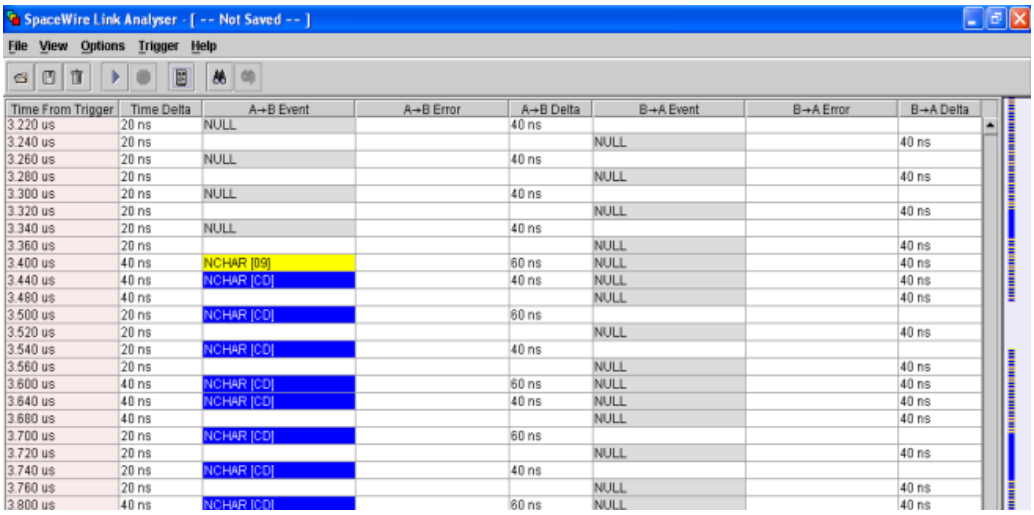


Figure 5: Display of packet level trace of analysed link activity.

The Link Analyser has enough memory to bi-directionally record 8,000 events. This can prove a limitation if a great deal of data needs to be analysed either side of a trigger event. Data filtering capabilities are provided in the Link Analyser to make maximum use of the available memory.

3 INTRODUCING THE SPACEWIRE LINK ANALYSER MK2

The SpaceWire Link Analyser Mk2, shown in Figure 6, combines the functionality of the SpaceWire Link Analyser and SpaceWire Monitor by providing both a USB 2.0 interface to a PC and ports suitable for interfacing to major logic analyser brands. It features two external trigger ports which can be independently configured as trigger in or out. Onboard memory provides enough capacity to store 16 million events.

Figure 6: The STAR-Dundee SpaceWire Link Analyser Mk2. The front panel (top) features, from left to right, 2 triggers and 2 SpaceWire ports. The rear panel (bottom) features, from left to right, 2 logic analyser ports, USB 2.0 connector and +5V power supply jack.

Drivers and software provide compatibility with Windows and Linux. The new software for the Link Analyser Mk2 builds on its predecessor by including the ability for the user to specify their own high-level protocols. Figure 7 shows a sequence of packets interpreted into the Remote Memory Access Protocol (RMAP).

Time Fro…	Time …	End A	End A D…	End B	End B Delta
0 ns		**(PID=1) Header:** RMAP Command			
		Path Address: 04			
		Target Address: FE			
		Instruction: Read Command			
		Increment Target			
		Key: 20			
		Reply Path: 00 00 01 03			
		Initiator Address: FA			
		Transaction ID: 1592			
		Extended Address: 00			
		Address: 00000001			
		Data Length: 000004			
4.080 µs	4.080 µs	**EOP** (4.080 µs @ 5.147 Mbytes/s)	4.080 µs		
5.620 µs	1.540 µs			**(PID=1) Header:** RMAP Reply	
				Path Address: 03	
				Target Address: FA	
				Instruction: Read Reply	
				Increment Target	
				Status: Success	
				Initiator Address: FE	
				Transaction ID: 1592	
				Data Length: 000004	
				Data:	
				00 00 00 0D	
8.520 µs	2.900 µs			**EOP** (2.900 µs @ 6.207 Mbytes/s)	2.900 µs
1.0046 ms	996.08…	**(PID=2) Header:** RMAP Command	1.00052 …		
		Path Address: 04			

Figure 7: Interpreting a recorded sequence of packets into RMAP.

4 CONCLUSION

This short paper has given a brief insight into the importance of link analysis and an overview of the SpaceWire Monitor and SpaceWire Link Analyser. The SpaceWire Link Analyser Mk2 unites the respective PC and logic analyser interfaces of these devices. It builds on these features by including 2,000 times the memory of the Link Analyser, a pair of external triggers and an enhanced version of software based on that of its predecessor. The SpaceWire Link Analyser Mk2 is a powerful, flexible tool for testing, validating and verifying a SpaceWire system.

5 REFERENCES

1. ECSS, Standard ECSS-E-ST-50-12C, "SpaceWire – Link, Nodes, Routers and Networks", European Cooperation for Space Standardization, July 2008.

2. S. Parkes, C. McClements, S. Mills and I. Martin, "SpaceWire: IP, Components, Development Support and Test Equipment", Proceedings of DASIA 2003, Prague, Czech Republic, 2-6 June 2003.

3. STAR-Dundee, "SpaceWire Link Analyser, http://www.star-dundee.com

OBSERVING AND TESTING SPACEWIRE THROUGH PCI AND PCI EXPRESS

Session: SpaceWire Test and Verification

Short Paper

Paul E. McKechnie

STAR-Dundee Ltd, c/o University of Dundee, School of Computing,

Dundee, DD1 4HN, Scotland, UK

Steve Parkes

University of Dundee, School of Computing, Dundee, DD1 4HN, Scotland, UK

E-mail: paul@star-dundee.com, sparkes@computing.dundee.ac.uk

ABSTRACT

When testing, validating and verifying systems, monitoring hardware can be used to observe and stimulate the components comprising the system under test. STAR-Dundee is developing the next generation of the SpaceWire PCI board and a SpaceWire PCI Express board to observe and stimulate up to four independent interfaces that implement the SpaceWire protocol. This paper describes the architectures of the Spacewire PCI-3 board and the SpaceWire PCI Express board. It also briefly describes the associated host software and outlines the benefits of these architectures.

1 INTRODUCTION

Effective testing requires the ability to accurately monitor the operation of the design under test. Within the SpaceWire application domain, there are several products available for monitoring and stimulating the links that comprise a SpaceWire network. This paper describes two boards that complement the existing products from STAR-Dundee and provide additional functionality to aid the designer while debugging their system. The boards being developed by STAR-Dundee are the next generation SpaceWire PCI board and the SpaceWire PCI Express board [1, 2].

2 SPACEWIRE PCI-3

The STAR-Dundee SpaceWire PCI-2 interface card is a test and development platform for the SMCS FPGA device provided by Astrium [3]. The SMCS is designed to be a general purpose interface device but is not suitable for some applications which require high performance and responsiveness between the software running on the PC and the SpaceWire links on the device.

The new STAR-Dundee SpaceWire PCI-3 interface, based on the existing hardware platform, provides new firmware which greatly improves the data transfer capabilities between the host system's software and the SpaceWire interfaces. As illustrated in Figure 1, the PCI Interface provides four bi-directional channels which are directly

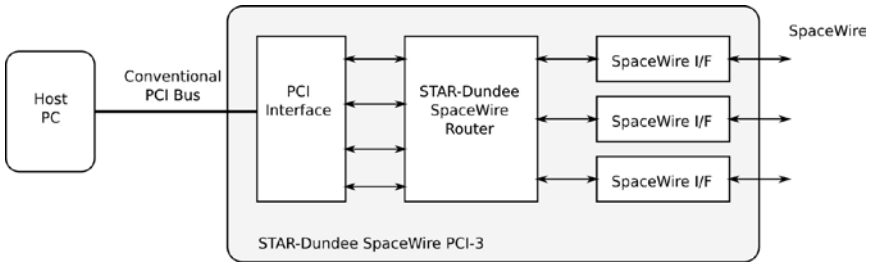


Figure 1: Architecture of the SpaceWire PCI-3 board

available to software running on the host PC. Each channel has individual buffering for efficient data transfer. A SpaceWire Router is also present, which adds a routing capability to the PCI card. Packets which are received on a SpaceWire port can be routed to another port or to the PCI interface. The SpaceWire router can also be configured to be bypassed by the SpaceWire interface so the card becomes an easy to use SpaceWire node with 3 interfaces. The extensive configuration capabilities of the SpaceWire router are available in both modes.

3 SPACEWIRE PCI EXPRESS

The new SpaceWire PCI Express board supports up to four SpaceWire links and consists of a SpaceWire router, on-board memory, a packet generator, a packet sink, multiple link monitors and upgradeable firmware, as illustrated in Figure 2. The board also employs a Direct Memory Access (DMA) engine for transferring data to the host computer. The maximum theoretical signalling rate for PCI Express is 2.5Gbits per second per lane for a 1st generation interface, which translates to 250Mbytes per second of transmitted data. However, the actual achievable data rates will be less due to the protocol overhead and software response time.

The SpaceWire router provides four external ports and several internal interfaces in addition to the configuration interface. The four external ports are fully compliant with the SpaceWire standard [4] and provide direct communication over such links. The internal interfaces provide access to the on-board memory, the host computer and the configuration interface. The on-board memory is accessible over SpaceWire via a hardware packet sink, which records all packets written to it. The host computer can access the board via a device driver, which provides the interface between the host computer and the SpaceWire ports. The internal interfaces are also accessible over the SpaceWire network via the router.

The on-board memory provides storage for all of the link monitors and the packet sink, which allows link activity to be recorded and statistical information to be gathered. The memory also contains an interface to the PCI Express bus so that data can be transferred to the host computer independently from the SpaceWire router. The link monitors can record many events as megabytes of memory are available providing significant temporal visibility.

The packet generator is connected to the SpaceWire router and can send packets to any interface on the router. It is therefore capable of interacting with any component in the SpaceWire network. The packet generator shares some of the functionality provided by the STAR-Dundee Conformance Tester [5] and is compatible with the SpaceWire Validation Software [6]. For example, the packet generator can transmit

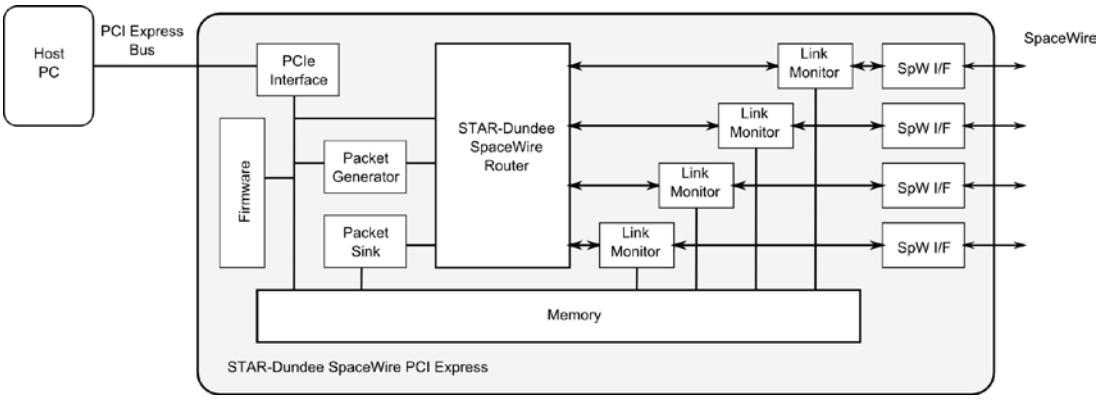


Figure 2: Architecture of the SpaceWire PCI Express board

predetermined packets or random data, while scheduling packets for transmission at specific times.

The packet generator can write packets to any interface on the SpaceWire router, which allows the packet generator to write to the packet sink or the connection to the host computer and its associated software driver. This capability is useful for debugging software applications where the rate of incoming data affects the performance and behaviour of the software application.

The packet sink is a SpaceWire interface to the on-board memory, which allows packets to be routed from any of the interfaces available on the SpaceWire router. Although the interface will block due to multiple packets accessing the sink simultaneously, there are no blocking delays caused by software. The operating system of the host computer and its application software are not involved in the operation of the packet sink and do not restrict the ability to receive packets.

The packet sink complements the functionality found in the SpaceWire Validation Software. For example, the sequence number and checksum included within a packet can be verified as they are received. A set of statistics showing the rate at which packets are received and any errors encountered are also made available to the host computer.

The link monitors observe both directions of each external SpaceWire interface while performing two key functions. First, they record packet activity on each of the external SpaceWire interfaces. Second, they capture statistical information from the traffic observed on each external SpaceWire interface. This statistical information includes packet transmission rates, link level events and observed errors.

The link monitors present on the SpaceWire PCI Express board contain a subset of the functionality found in the STAR-Dundee SpaceWire Link Analyser but still provide many features that are beneficial for monitoring and debugging SpaceWire applications [7]. For example, the link monitors record packets entering and leaving the SpaceWire router and can highlight the exit path of each packet. This functionality could be used to verify that packets are being routed correctly.

The PCI Express board uses firmware that can be upgraded in the field. The firmware can provide additional functionality for the SpaceWire router, packet generator, packet sink and link monitors. Most importantly the firmware will be upgradable over

the PCI Express interface, which allows the firmware to be updated from any location where the host computer is accessible.

4 HOST SOFTWARE

The SpaceWire PCI Express and SpaceWire PCI-3 devices will have drivers for both Windows and Linux operating systems and will use the new STAR-Dundee driver architecture and API. The new API allows applications to work with any of the STAR-Dundee devices so that applications could potentially work with SpaceWire PCI Express, SpaceWire PCI-3, SpaceWire Router-USB and SpaceWire-USB Brick devices. The new API also provides reusable components to provide features such as Plug and Play (PnP) and RMAP targets and initiators, so the PCI devices can respond to PnP requests [8], for example. STAR-Dundee's existing applications will work with both PCI devices.

5 CONCLUSION

This paper has described the architecture of the next generation SpaceWire PCI and SpaceWire PCI Express boards being developed by STAR-Dundee. The benefits of both boards have been presented and the enhanced test capabilities of the SpaceWire PCI Express board have been described.

6 REFERENCES

1. PCI-SIG, "PCI Local Bus Specification", March 2002, http://www.pcisig.com/specifications/conventional/

2. PCI-SIG, "PCI Express Base Specification Revision 2.0", 20th December 2006, http://www.pcisig.com/specifications/pciexpress/specifications

3. STAR-Dundee, "SpaceWire PCI-2", August 2007, available from http://www.star-dundee.com/

4. ECSS, "SpaceWire – Links, nodes, routers and networks", ECSS-E-ST-50-12C, July 2008, available from http://www.ecss.nl.

5. S. Parkes and M. Dunstan, "Debugging SpaceWire Devices Using the Conformance Tester", International SpaceWire Conference 2007, Dundee, Scotland, UK, September 2007

6. STAR-Dundee, "SpaceWire Validation Software", April 2008, available from http://www.star-dundee.com/

7. STAR-Dundee, "SpaceWire Link Analyser", August 2009, available from http://www.star-dundee.com/

8. S. Mills, C. McClements, S.M. Parkes, "STAR-Launch and Network Discovery", International SpaceWire Conference 2010, St. Petersburg, Russia, 22-24 June 2010.

SPACEWIRE TEST AND DEMONSTRATION UTILISING THE INTEGRATED PAYLOAD PROCESSING MODULE

Session: SpaceWire Test and Verification

Short Paper

Walter Errico, Pietro Tosi

CAEN AURELIA SPACE srl, Viareggio, Italy

Jørgen Ilstad, David Jameux

European Space Technology Centre, Noordwijk, Netherlands

Riccardo Viviani, Daniele Collantoni

ABSTRAQT srl, Lucca, Italy

E-mail: Jorgen.ilstad@esa.int, david.jameux@esa.int, w.errico@caen.it, p.tosi@caen.it, r.viviani@abstraqt.it, d.collantoni@abstraqt.it

ABSTRACT

A simplified On-Board Data Handling system has been developed by CAEN AURELIA SPACE and ABSTRAQT as PUS[1] over SpaceWire demonstration platform for the Onboard Payload Data Processing laboratory at ESTEC. The system is composed of three Leon2 based IPPM (Integrated Payload Processing Modules) computers that play the roles of Instrument, Payload Data Handling and Satellite Management units. Two PCs complete the test set-up simulating an external Memory Management and the Ground Control units. Communication among units take place primarily through SpaceWire links, RMAP[2] protocol is used for configuration and HK. A limited implementation of ECSS-E-70-41B Packet Utilisation Standard (PUS) over CANbus and MIL1553 has been also realized. The Open Source RTEMS has been running on the IPPM AT697E CPU as RTOS.

IPPM DEMONSTRATION SYSTEM OVERVIEW

The demonstration platform has been designed in order to emulate a simplified On-Board Data Handling System. It consists of three equivalent IPPM and two personal computers. The demonstration architecture is depicted in Figure 1, where a Windows based PC acts as a Ground Control Unit (GCU), while a second PC simulates an external Mass Memory Unit (MMU) and each IPPM module emulates the different components of the on board system:

- An Instrument Unit (IU) plays the role of a scientific instrument that generates data to be processed by the payload data handling unit. The IU simulates an on-board camera and generates raw images data and supports a limited set of PUS services, which allow starting and stopping images data generation.
- A Payload Data Handling Unit (PDHU) manages instrument data incoming from the Instrument Unit. Data are stored by PDHU into a Memory Management Unit (MMU) simulated by a Windows based PC. The PDHU is able to store the data in

both raw and compressed format (JPEG). Compressed data are downloaded to the Ground Control Unit after request.

- The whole IPPM on-board system is managed by a Satellite Management Unit (SMU). This module sends and receives messages to and from the ground control unit and the other IPPM boards. The SMU provide a delayed Telecommands scheduling service and it is in charge to distribute a time synchronization signal to the other boards by means of SpaceWire Time-Codes

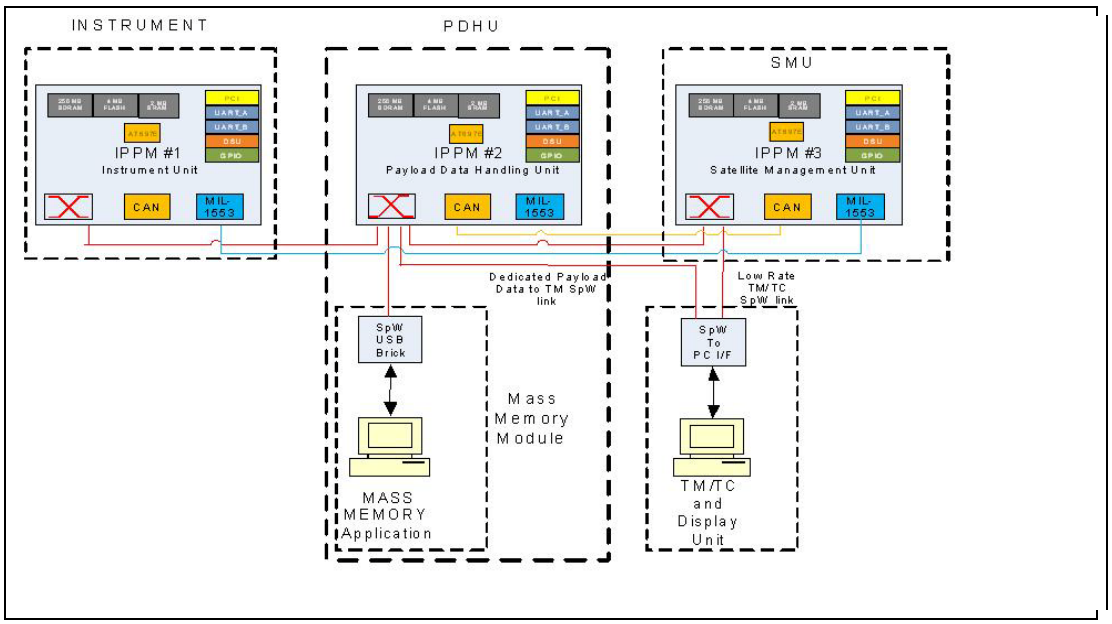


Figure 1: On-Board Data Handling System demonstration architecture

The three IPPM boards involved in the system are custom low power computers based on LEON architecture and running RTEMS as operating system. Each board has a large amount of on-board memory and wide networking resources such as high speed SpaceWire links and, in addition, CAN and MIL-STD-1553 bus interfaces designed to distribute low and medium rate command and control signals.

In the frame of the activity a portable SW library has been developed which supports RMAP transactions and a subset of PUS services. The library provides a platform independent interface to the underlying hardware. All of the three boards run both Housekeeping and Time Synchronization processes: the first process deals with measuring and collecting information about the status of the system, while the second has been designed in order to maintain the clock of both the IU and the PDHU synchronized with the clock of the SMU. The IU runs a camera emulator process that transfers raw image data to the PDHU using PUS Large Data Transfer service. The PDHU stores the received data into the MMU exploiting the RMAP features provided by the software library. An Off-line compression process is in charge to compress the raw images into a dedicated channel of the MMU, while a Data Downlink process download the compressed images to the ground control unit where can be displayed through the GUI. The SMU has a direct SpaceWire link to the ground control unit.

INSIDE THE INTEGRATED PAYLOAD PROCESSING MODULE

The Integrated Payload Processing Module (IPPM) is a single module self-contained computer based on RISC CPU, equipped with a large amount of memory on-board and having a wide inter-networking capability.

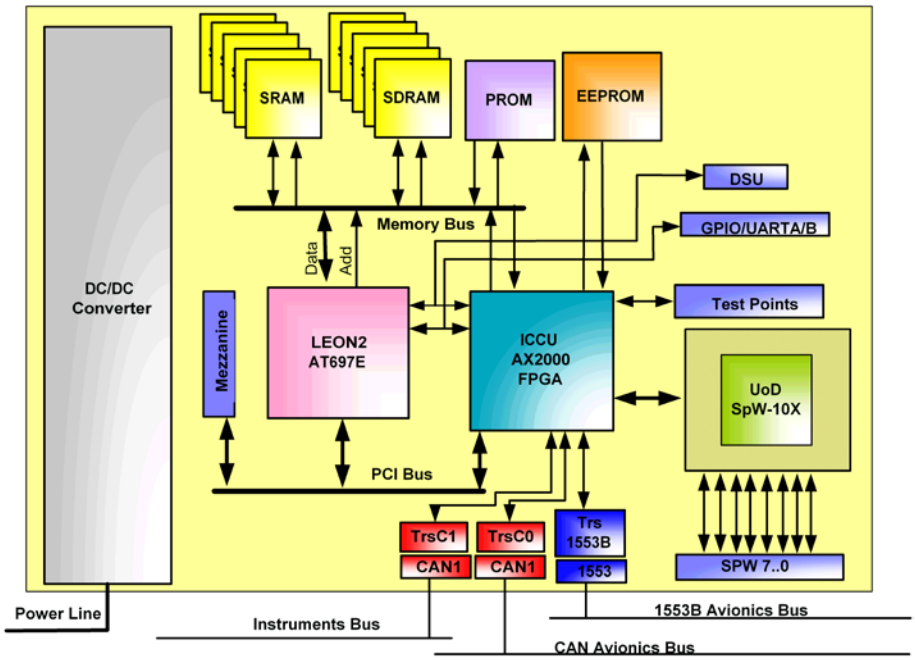


Figure 2: IPPM block diagram

The LEON2 CPU controls the PROM, SRAM and SDRAM components through its memory bus that is also connected to the ICCU FPGA. The ICCU, that internally bridges the memory bus to the FLASH interface, keeps the ownership of the FLASH interface to manage the on board programmability. Data exchange between LEON2 and ICCU takes place on the PCI bus, while ICCU and UoD_SpW-10X router communicate through the data dedicated ports.
The wide IPPM inter-networking is guaranteed by 8 SpW links, connected to the UoD_SpW-10X router, two different CAN buses and a MIL-STD-1553B peripheral block capable of 1553 Bus Controller, Remote Terminal or Bus Monitor functionalities.

ARCHITECTURE OF THE IPPM SOFTWARE LIBRARY

The IPPM software library provides an object-oriented abstraction for both PUS and RMAP standards. The library is organized in different layers to provide a uniform platform-independent interface to the underlying hardware or operating system. The layers are described below, starting from the lower level layer up to the upper level.

The **Communication D evice Layer** directly manages the communication devices. This layer is designed to send and receive data using a specific communication device (SpaceWire, CAN, MIL-1553, etc) and, in principle, makes the upper layers independent with respect to the communication device used to transfer the data.During the development phase various tests concluded that transferring large data sets using CAN/1553 became impractical, thus these buses have been used for transferring HK parameters to the SMU.

The **CoDec Layer** is in charge to encode outgoing messages from the routing layer to the communication layer, and decode incoming messages from the communication layer to the routing layer.

The **Routing Layer** makes the upper layer independent of routing strategies and dispatching methods. While static configuration tables have been utilized for CANbus/1553 topologies, the logic addressing with the 10X-Router adaptive routing capabilities have been exploited for the SpaceWire links.

The **Message Layer** deals with data transformation: application data and requests are encapsulated into objects which reflect the structure of a standard message, such as Telecommand and Telemetry messages of the Packet Utilization Standard protocol (PUS). Telemetry and Telecommand messages are implemented according to the On-Board Software Framework [3], with additional PUS services and RMAP features.

The **Application Layer** decouples a generic Application Processes from the lower layer providing them a standard communication interface.

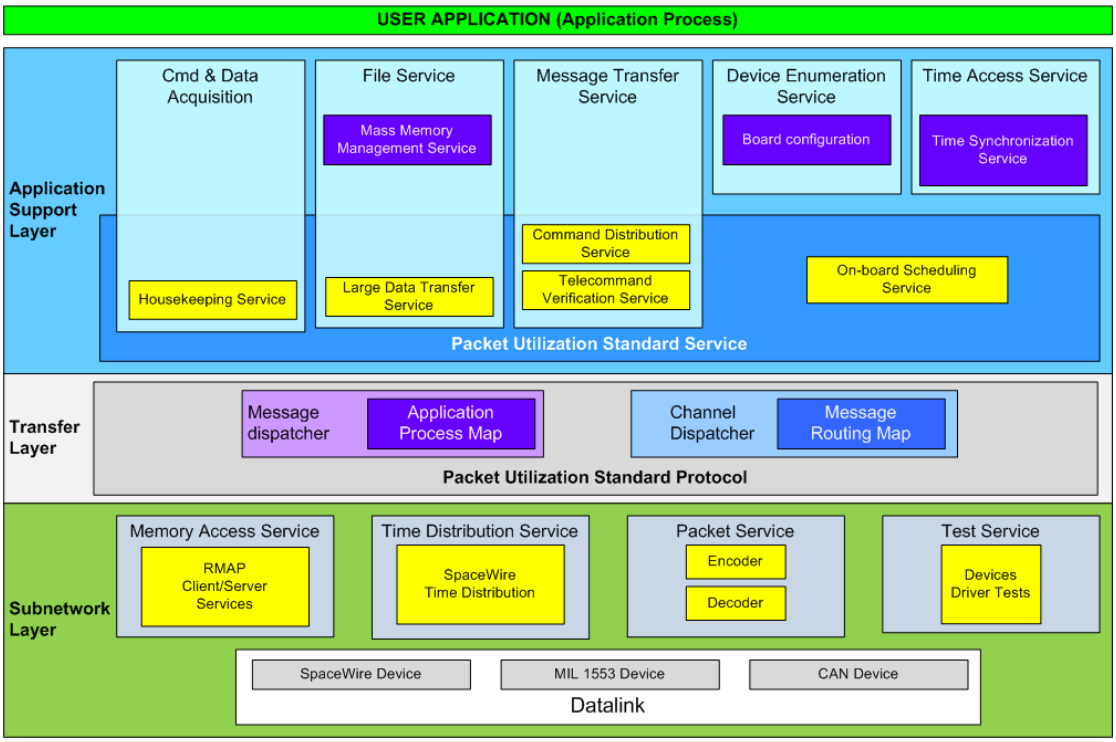


Figure 3: Main modules of the IPPM software library

The current version of the software library is organized in modules, as depicted in Figure 3, and supports Telecommand Verification, Housekeeping, Large Data Service, Command Distribution and On Board Scheduling standard PUS services together with RMAP capabilities; while SpaceWire, MIL-1553 and CAN are the currently supported communication devices.

REFERENCES

1. ECSS, E-70-41 Telemetry & Telecommand Packet Utilization Standard.

2. ECSS, E-ST-50-52C SpaceWire - Remote Memory Access Protocol (normative).

3. Alessandro Pasetti and Vaclav Cechticky, Automatic Control Laboratory of ETH-Zurich, *On-Board Software Framework*, http://www.pnp-software.com/ObsFramework.

PREPARING THE RASTA SOFTWARE FOR SPACEWIRE BACKPLANES

Session: SpaceWire Test and Verification
Short Paper

Daniel Hellström, Kristoffer Glembo, Sandi Habinc

Aeroflex Gaisler, Kungsgatan 12, SE-411 19 Göteborg, Sweden

daniel@gaisler.com, kristoffer@gaisler.com, sandi@gaisler.com

ABSTRACT

Replacing a Peripheral Component Interconnect (PCI) [3] backplane with a SpaceWire [1] backplane would normally require major changes to hardware and software. Today most PCI boards [6] used in the Reference Avionics System Test-bench Activity (RASTA) already include SpaceWire interfaces and connectors on the front panel. Aeroflex Gaisler has developed new abstract bus models for the *Driver Manager* (DM) used previously in RASTA software [5] solutions, in order to simplify porting of existing and newly developed RTEMS [4] software and improve code reuse. This paper focuses on the software aspects of this development.

BACKGROUND

The RASTA facility is a test-bedding element of the avionics laboratory at ESA [6]. It provides a representative environment for demonstration, evaluation and testing of hardware, software and communication protocols that comprise on-board data systems. RASTA has since grown in to a standardized platform [10] for on-board data system developments and is made mandatory in many ESA activities. RASTA consists of on-board computers, boards on-board I/O boards, TM/TC boards, etc.

Most of the RASTA hardware elements are implemented as system-on-chip designs based on the AMBA [9] on-chip bus. The processor element is the LEON2 or LEON3 SPARC processors, residing on the AMBA on-chip bus. The processor is connected via AMBA to one or more cores implementing various interfaces to buses such as PCI and SpaceWire. Other elements, such as on-board I/O, are also based on the AMBA on-chip bus, featuring interfaces such as PCI, SpaceWire, CAN, Mil-Std-1553B, CCSDS/ECSS Telemetry (TM) &Telecommand (TC) etc. Thus each board comprises an ASIC/FPGA that implements a system-on-chip based on the AMBA on-chip bus.

The PCI bus is currently used as the main interface between these boards. This paper discusses the next step, where PCI is replaced with SpaceWire links for implementing communication between boards. Communication between the processor and any other resource can be done in several steps; firstly the processor accesses local resources (i.e. cores) over the AMBA bus; secondly it accesses other units via PCI (current baseline); thirdly it accesses other units via a SpaceWire link (discussed in the paper).

RMAP INITIATOR STACK

Throughout this paper all SpaceWire accesses involve Remote Memory Access Protocol (RMAP) [2] commands sent to and responses received from RMAP targets. The RMAP header and CRC are generated by an *RMAP initiator stack* developed by

Aeroflex Gaisler (CRC is generated by hardware where supported), received RMAP responses are also processed by the stack. All RMAP commands are generated according to the ECSS standard [2]. The initiator stack supports path addressing, logical addressing and all defined transfer types. The initiator stack does not implement the RMAP command/response transportation itself, but relies on an *RMAP stack driver*, which in turn relies on SpaceWire packet transportation implemented by an underlying *SpaceWire driver*. Layering the software makes it possible to support SpaceWire cores from different vendors. The initiator stack does not require any particular RMAP support in hardware since the RMAP commands are generated in software. An *RMAP stack driver* has been implemented supporting the RTEMS SpaceWire drivers for the GRLIB GRSPW/GRSPW2 IP cores [7][8].

DRIVER MANAGER

1.1 OVERVIEW

Device drivers rely on services such as Plug&Play scanning and IRQ (Interrupt Request) management. The services are often implemented differently for different bus types (e.g. AMBA or PCI), and as a result different Application Programming Interfaces (API) are utilized when accessing these services. The *Driver Manager* (DM), developed by Aeroflex Gaisler, in RTEMS provides services such as driver loading (finding devices and appropriate drivers for them), IRQ handling etc. In an attempt to provide a single API for the above mentioned services regardless of bus type, the DM was created and abstract models for buses, devices, bus drivers and device drivers were introduced. These concepts have been successfully implemented and used in RASTA for on-chip AMBA buses, PCI buses and *AMBA-over-PCI* (i.e. communication from local-AMBA bus over PCI to remote-AMBA bus).

This paper introduces two new bus models; the *SpaceWire Network* and *AMBA-over-RMAP* (i.e. communication from local-AMBA bus over SpaceWire to remote-AMBA bus). In order to support the new models the DM API has been extended with READ and WRITE operations that can be used for communication over all bus types.

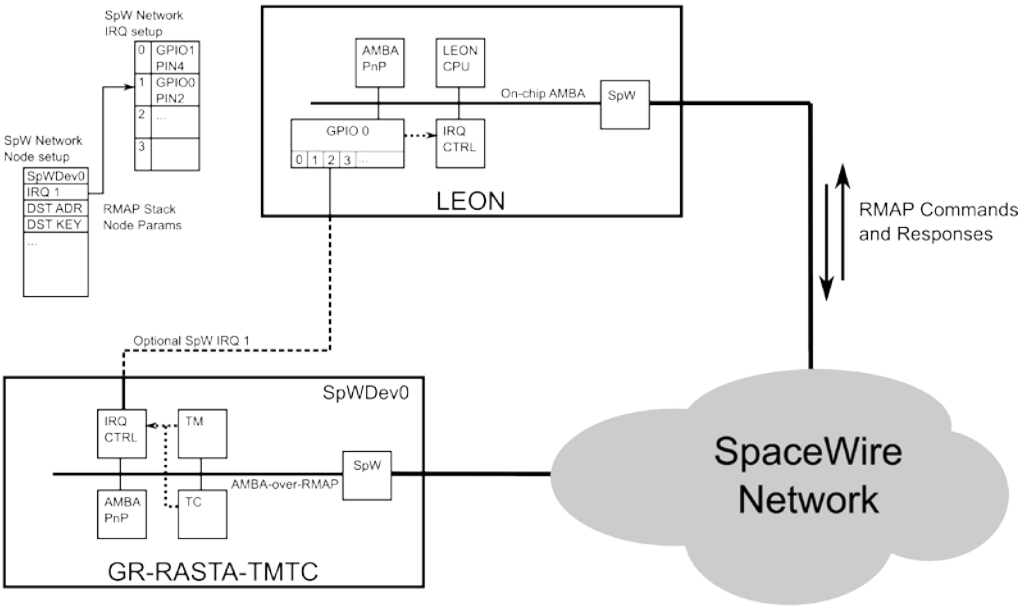


Figure 1: System overview

1.2 SPACEWIRE NETWORK BUS MODEL

The *SpaceWire Network* model implements IRQ management and read and write access to SpaceWire RMAP targets. The services are used by *SpaceWire Node drivers*. The SpaceWire standard currently does not define Plug&Play, instead the SpaceWire network configuration including all nodes are provided by the user. The most important node configuration parameters are the SpaceWire destination address, destination key and a virtual SpaceWire network IRQ number.

In each service request initiated by a *SpaceWire Node driver*, a pointer identifying the particular node is given to the *SpaceWire Network* model. From the node configuration the parameters used in lower layers can be determined. Read and write access requests are for example implemented by adding SpaceWire destination address/key and passing on the request to the RMAP initiator stack.

SpaceWire currently does not transport IRQ over the link. IRQ is optional and when enabled are communicated as discrete level sensitive signals (requiring some sort of GPIO hardware). The *SpaceWire Network* bus model works with virtual SpaceWire IRQ numbers that can be resolved into a GPIO pin accessible by the processor. In order to determine what caused the IRQ and to acknowledge the source, the RMAP target must be accessed over the SpaceWire bus.

There are a number of reasons why an RMAP request cannot be performed in an interrupt context, for example the time for completing an RMAP request may be too long or the initiator stack itself may rely on interrupts. Instead of acknowledging the IRQ source directly the GPIO interrupt is masked, stopping further IRQs from the source, and a high-priority Interrupt Service Routine (ISR) task is awoken. The task is responsible for calling all registered ISRs, the IRQ source will finally be acknowledged and before suspending the ISR task will unmask the GPIO interrupt.

1.3 AMBA-OVER-RMAP BUS MODEL

The *AMBA-over-RMAP* bus model provides AMBA device drivers with IRQ management and read and write access to AMBA cores over SpaceWire. The model can be seen as a generic *SpaceWire Node* driver laying on top of the *SpaceWire Network* model. By reusing existing AMBA bus layers and Plug&Play scanning, the footprint is reduced, and the *AMBA-over-RMAP* bus model interface is similar to a standard on-chip AMBA bus model, with a few additions.

The IRQ service is implemented on top of the *SpaceWire Network* IRQ service, the constraints are thus propagated to *AMBA-over-RMAP* device drivers. ISRs are executed in a high-priority task context in order to read and write the AMBA bus, this differs from the on-chip AMBA bus model.

Critical areas, such as accessing registers, in combination with changing the state of the software, are normally protected from an ISR by changing the Processor Interrupt Level (PIL) to disable interrupts during the execution of the critical section. *AMBA-over-RMAP* drivers cannot change the PIL to protect themselves, since a register access over SpaceWire may require IRQ or may take substantial time to complete. The same applies to the ISR itself that now executes in task context, it may be pre-empted while accessing an AMBA register over SpaceWire. This problem is solved by protecting critical areas with semaphores and introducing semaphores in the ISRs.

Even though register and memory accesses differ radically from the on-chip AMBA bus model, the simple API provided minimizes the required changes. The driver author needs not to worry about RMAP parameters as they are added in lower layers. The major differences are that accesses can be pre-empted by other tasks while waiting for response (depending on initiator stack, SpaceWire driver and scheduling policy) and that accesses may fail due to SpaceWire communication errors.

FUTURE IMPROVEMENTS

The current implementation of the *RMAP initiator stack* and the *SpaceWire driver* may create lock congestion when multiple tasks access the SpaceWire link simultaneously. Setting up task scheduling may reduce the problem significantly, however it cannot be avoided completely. Current software could be improved and support for multiple DMA channels could be added for the GRSPW2 core to avoid lock congestion.

Error handling could be improved in the bus models. RMAP read and write accesses may fail, layers beneath the AMBA device driver involved could be made to handle failures or re-execute requests automatically. With the bus model, error handling can be isolated per SpaceWire network, per SpaceWire node or per AMBA device.

CONCLUSIONS

The SpaceWire backplane software is already used in a separate ECSS TM/TC FPGA [12] project to set up, control and transport frames [11] over SpaceWire using RMAP.

The abstract bus models of the Driver Manager (DM) makes is possible to reuse and maintain most of the interface towards an AMBA driver, regardless of bus location (on-chip or over SpaceWire). Even though this paper describes a number of changes

that AMBA drivers must undergo for conversion to an *AMBA-over-RMAP* bus model, they are relatively few and easy and the driver structure can be maintained. Nevertheless, most importantly the user interface remains the same.

Currently SpaceWire Plug&Play is the only missing link in order to go completely Plug&Play, all the way from the controlling processor to a resource on the other side of a SpaceWire network.

REFERENCES

1. SpaceWire - Links, Nodes, Routers and Networks, ECSS-E-ST-50-12C
2. SpaceWire - Remote memory access protocol, ECSS-E-ST-50-52C
3. PCI Local Bus Specification, Revision 2.3, 2002, www.pcisig.com
4. RTEMS, Real-Time Executive for Multiprocessor Systems, www.rtems.org
5. RASTA Interface Control Document (ICD) - Software, TEC-EDD/2007.32/GF
6. RASTA Interface Control Document (ICD) - Hardware, TEC-EDD/2007.31/GF
7. GRLIB IP Library User's Manual, Aeroflex Gaisler, www.gaisler.com
8. GRLIB IP Core User's Manual, Aeroflex Gaisler, www.gaisler.com
9. AMBA Specification, Rev 2.0, ARM IHI 0011A, Issue A, ARM Limited
10. S. Habinc, *Crucial SpaceWire Elements in RASTA*, ISC 2008
11. S. Habinc, *CCSDS/ECSS Telemetry and Telecommand for RASTA*, DASIA 2009
12. S. Habinc, *ECSS TM-TC Component with SpaceWire RMAP Interfaces*, ISC 2010

TOPNET EVOLUTION

Session: SpaceWire Test and Verification

Short Paper

Raffaele Vitulli

European Space Agency – Keplerlaan 1, 2201AZ Noordwijk -The Netherlands

E-mail: Raffaele.Vitulli@esa.int

ABSTRACT

The Virtual Spacecraft Integration (a.k.a. "TopNET") is achieved through the use of a network such as the Internet. To determine the benefits and limitations of virtual spacecraft integration, ESA conducted a pilot study involving spacecraft and equipment manufacturers in different countries across Europe. They conducted experiments using the SpaceWire Internet Tunnel device to remotely integrate components and reported back on their findings. Their results were very positive. Despite the identified benefits, there are potentially some limitations of virtual spacecraft integration. In the paper, the outcome of the pilot study will be shortly summarized, and a strategy to enhance "synchronous" communication will be presented. Moreover, a further objective is to extend the TopNET concept, having in mind the CCSDS SOIS architecture.

1 INTRODUCTION

The concept of virtual spacecraft integration has been described in previous papers by ESA/University of Dundee [2]. It provides a means by which integration and testing of spacecraft components can be performed without the need to bring each of the components to one physical location. The SpaceWire standard [1] aims to improve reusability, promote compatibility and reduce system integration costs. Virtual spacecraft integration has the potential to reduce system integration costs still further, by reducing travel and by identifying problems at an earlier stage of spacecraft development than is currently the case.

Virtual integration is achieved through the use of a network such as the Internet. A section of the spacecraft's onboard bus is replaced with a virtual connection over the network, allowing components to communicate with one another, despite potentially being great distances apart.

2 THE SPACEWIRE INTERNET TUNNEL

The SpaceWire Internet Tunnel is a tool for performing virtual spacecraft integration in a SpaceWire network and it has been originally developed by the University of Dundee under ESA contract. An example SpaceWire network which could benefit from virtual spacecraft integration is shown in Figure 1. This network contains two

separate sub-systems, which may be developed by different companies, possibly in different countries. This is quite common in European missions, for example.

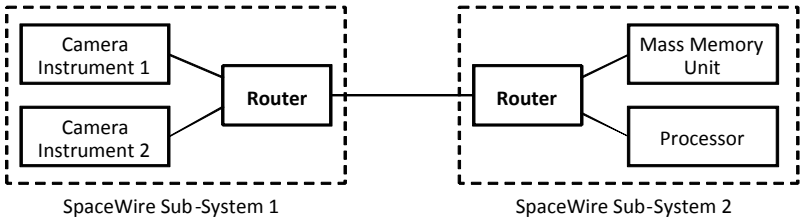


Figure 1: Example SpaceWire network containing two distinct sub-systems

A SpaceWire Internet Tunnel replaces a SpaceWire link in an onboard network, and consists of both software and hardware components. A SpaceWire cable representing one end of the link to be replaced by the SpaceWire Internet Tunnel is connected to a SpaceWire IP-Tunnel device. This device is then connected to a PC by a USB cable. Software running on the PC manages the Tunnel and allows traffic crossing the Tunnel to be monitored and recorded. A similar set-up is used at the other end of the link being replaced, and the software running on the two PCs tunnels traffic received on the SpaceWire links over a network to the other end. This arrangement is shown in Figure 2, where the two sub-systems from the example network in Figure 1 have been connected virtually using a SpaceWire Internet Tunnel. These two sub-systems may be in the same lab, or may be in different continents.

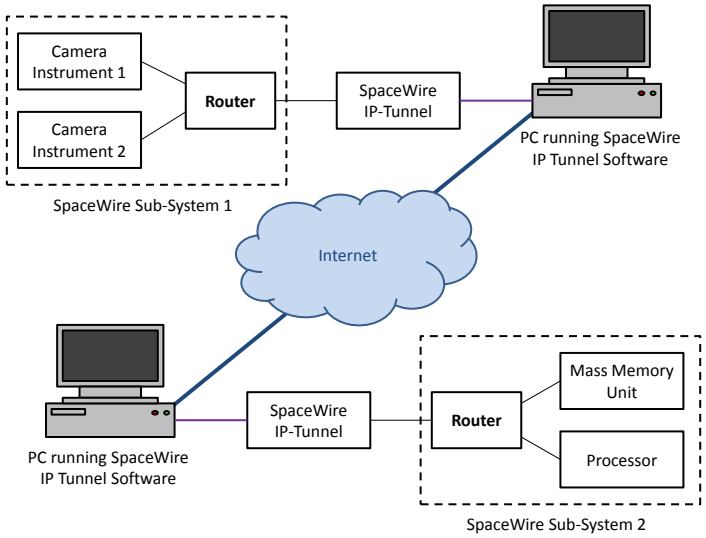


Figure 2: Example SpaceWire network containing two sub-systems integrated virtually

As well as exchanging data packets, the Tunnel also ensures that the link state is reflected at each end of the Tunnel. This means that if a link is disconnected at one end of the Tunnel, the link at the other end will also be disconnected. Other than the increased latency and reduced bandwidth, the Tunnel is almost transparent.

3 THE TOPNET PILOT STUDY: BENEFITS AND LIMITATIONS

An ESA funded pilot study, the "TopNet Pilot Study", was implemented to investigate the benefits and limitations of virtual spacecraft integration when in use within a real project environment. The study involved three consortia, each consisting

of partners spread across Europe. Each consortium proposed experiments where SpaceWire devices situated in each of the consortium's partners would be virtually integrated.

On completing their experiments, each consortium presented their findings. They also compiled a report containing their results and conclusions. The general consensus of those involved in the pilot study is that although there is still the possibility to make improvements, virtual spacecraft integration and the SpaceWire Internet Tunnel is a very useful tool for performing integration testing. Virtual spacecraft integration has the potential to reduce costs and time in spacecraft development, while also improving the completed product. By allowing integration testing of spacecraft components to be performed at an earlier stage, problems in interfaces and at the application level, for example, can be identified and corrected earlier, which reduces the time and money required to address such issues. Travel can also be reduced through use of virtual spacecraft integration as components can be integrated without bringing them to a single physical location. This has an environmental benefit in addition to a financial one.

Cooperation between organisations, or between sites within one organisation, can also be improved through use of virtual spacecraft integration. Engineers working on different subsystems, who may not normally have contact with one another until the integration phase late in the development cycle, may instead communicate at an earlier stage in the development cycle. Virtual spacecraft integration also provides much more flexibility than traditional integration procedures. The time at which integration testing is performed is much more flexible, as devices do not all need to be in the same place at a specific date and time. It is also easy to replace components with simulators when performing virtual integration testing, while test and analysis equipment can also be integrated virtually.

Despite these benefits there are potentially some limitations of virtual spacecraft integration. Use of a network connection such as the Internet introduces limits in bandwidth and latency. Neither are guaranteed in Internet communication, and both bandwidth and latency can vary greatly during a connection's lifetime. This limitation can affect cases when synchronous communication is required during the integration of two different modules.

Firewall restrictions can pose a problem to virtual spacecraft integration. In Internet communications there must be a server present. Such servers normally require special firewall permissions to be granted by a network administrator, and these permissions may not always be given to PCs running in a lab. Some organisations only allow certain traffic, such as web and FTP, to cross their firewall. The addition of a SpaceWire Internet Tunnel Server can address the firewall issue, where PCs cannot act as a server behind a company's firewall.

4 ENHANCING SYNCHRONOUS COMMUNICATION

Starting from the valuable feedback received by the pilot study, it seems necessary to improve the TopNET concept, in order to overcome the above-mentioned limitations. In particular, the latency can affect cases when synchronous, real time communication is required during the integration of two different modules.

For the Internet tunnelling communication, regular TCP/IP is used. It is well-known that TCP/IP implements a "best-effort" paradigm without guaranteeing latency. The reason for this is that queuing delays sum up at each router. TCP/IP is a "best effort" protocol and tries to fill the routers queues. Possible solutions in order to have a guaranteed latency are to use a customised communication protocol**:**

- TCP Westwood+ [3] is available in Linux kernel and it is a sender-side only modification of the TCP protocol stack that optimizes the performance of TCP. The use of TCP Westwood+ can provide benefits in terms of latency, since it is known to provide less queuing. In case of a particular scenario, i.e. when the bandwidth of the tunnel is known, as it is in corporate intranet, TCP Westwood+ can be further optimised to improve the performances.

- Design rate-based transport protocol at application level executed over the UDP, as it is done for applications that are time-sensitive, such as VoIP or Video Conferencing [4].

If the suggested modification for the Internet Tunnelling communication will bring the expected benefits and a guaranteed latency is obtained, new opportunities will be created for the Virtual Satellite Integration, where real-time communication will be possible. If the latency, other than been guaranteed, is also low enough, the TopNET concept could be extended to consider additional Data Link Layers besides SpaceWire, in order to cover the full CCSDS SOIS architecture [www.ccsds.org].

5 TopNET Evolution: Conclusions

In this paper, the outcome of the Pilot Study has been shortly summarized, and a strategy to enhance "synchronous" communication has been presented. The long term objective is to extend the TopNET concept, having in mind the CCSDS SOIS architecture. This means the introduction of additional Data Link Layers besides SpaceWire and the use of Quality of Service metrics. Future activities planned by ESA will pave the way to reach these objectives.

6 References

1. European Cooperation for Space Standardization, "SpaceWire, Links, Nodes, Routers and Networks", Standard ECSS-E-50-12A, Issue 1, European Cooperation for Space Data Standardization, February 2003.

2. S. Mills, S. Parkes, R. Vitulli, "The SpaceWire Internet Tunnel And The Advantages it Provides for Spacecraft Integration", International SpaceWire Conference 2008, Nara, Japan, November 2008.

3. A. Dell'Aera, L. A. Grieco and S. Mascolo, "Linux 2.4 Implementation of Westwood+ TCP with rate-halving: A Performance Evaluation over the Internet", IEEE International conference on Communication (ICC 2004), Paris, June 2004.

4. L.A. Grieco, S. Mascolo, "Adaptive Rate Control for Streaming Flows over the Internet", ACM Multimedia Systems Journal, Regular paper, Volume 9, Issue 6, pp. 517 -532, Jun. 2004.

STAR-DUNDEE VIRTUAL DEVICES AND SYSTEM SIMULATION

Session: SpaceWire Test and Verification

Short Paper

Stuart Mills, Alex Mason

STAR-Dundee, c/o School of Computing, University of Dundee, Dundee, Scotland, UK

Steve Parkes

University of Dundee, School of Computing, Dundee, Scotland, UK

E-mail: stuart@star-dundee.com, alex@star-dundee.com, sparkes@computing.dundee.ac.uk

ABSTRACT

STAR-Dundee's latest software API and applications allow users to build virtual SpaceWire networks on their PCs. Users can create their own virtual devices to represent hardware that may not yet be available, or to prototype features that aren't implemented in existing hardware. Virtual SpaceWire networks can then be constructed using these virtual devices. The virtual networks can also be integrated with one or more physical SpaceWire networks for testing, or can be tested in an entirely virtual environment.

This paper describes the functionality provided to allow users to create virtual devices, and the mechanisms used to integrate these virtual devices with one another and with physical SpaceWire networks. The advantages and potential limitations of this concept of virtual integration are identified and discussed.

1 INTRODUCTION

STAR-Dundee supplies a number of SpaceWire devices for test and development which can be connected to a PC [1]. Software is provided with these devices to allow users to send and receive traffic on a SpaceWire network, to configure and monitor devices on a SpaceWire network, and to perform many other tasks.

Users of STAR-Dundee equipment can also write their own applications to communicate with the SpaceWire devices and the SpaceWire network. An Application Programming Interface (API) is provided which allows software to be developed in C, C++, Java and many other languages.

Recently this API has been substantially improved, with new functions to make the development procedure much easier and to simplify common tasks. The new API also provides a number of additional features. The same API can be used to communicate with all STAR-Dundee devices, regardless of whether they are routers

or interfaces, and whether they are connected by USB, PCI, or some other mechanism.

The focus of this paper, however, is the API's support for virtual devices. Virtual devices are virtual representations of physical devices, created onboard the PC. These virtual devices can be routers or interfaces and are treated exactly like physical devices by the API.

2 USING VIRTUAL DEVICES

Virtual devices can be created using the API or through the STAR-Launch [2] software provided. STAR-Launch is a graphical tool which can be used for many purposes. As with normal devices, virtual devices can be configured using the API or STAR-Launch, and traffic can be sent to them or received from them.

Virtual devices can be connected together using virtual SpaceWire links, and applications and physical devices can also be connected together using the same virtual links. This allows a virtual SpaceWire network to be constructed inside a PC, with no physical hardware required. Alternatively virtual and physical devices can be combined in a single network. This allows a subsystem to be replaced with a virtual implementation, which may be useful in projects where subsystems are developed in different locations and/or organisations.

Using virtual devices also allows for additional debugging and status information to be made available. A virtual device can provide information to the user, which may not be possible with the physical implementation. Virtual devices can display error information on screen, or record statistics to file. This may not be possible in the physical implementation of these devices.

The use of virtual links also makes it much easier to analyse the traffic crossing a link. Virtual Link Analysers allow the traffic crossing a virtual SpaceWire link to be monitored. In a physical system, analysing a link is likely to involve replacing a cable with two cables and a SpaceWire Link Analyser [3]. In a virtual system, a Link Analyser can be added to a virtual link with the click of a button.

A planned improvement is the addition of the SpaceWire IP Tunnel [3], a tool for virtual spacecraft integration [4, 5], which allows subsystems to be connected virtually over the Internet. This would allow virtual subsystems to be integrated virtually, and would mean that a group working on one subsystem could debug problems with their subsystem, while connecting remotely to the other subsystems. These subsystems could in turn be monitored and debugged by their developers.

3 VIRTUAL ROUTING

A SpaceWire network is made up of links, nodes and routers [6]. To connect together the nodes in a large virtual network, in addition to the virtual links, virtual routers are required.

The STAR-Dundee API includes a Virtual SpaceWire Router, which can be used for a number of purposes, including the basic task of routing traffic around a virtual network. The Virtual SpaceWire Router can be used to test different router

configurations, e.g. different timeouts, different routing tables, etc. It can also be set to act as a time-code master, in order to test the behaviour of a network or device when time-codes are distributed across the network.

Virtual routers can also be useful in systems which aren't obviously using virtual components. For example, a PC might have two applications running, each of which wishes to receive packets sent to a device connected to the PC. The first application may wish to receive packets sent to logical address 251. The second application may wish to receive packets sent to logical address 252. To achieve this, a virtual router can be connected to the device, with packets with logical address 251 routed to the first application and packets with logical address 252 routed to the second application. The two applications can now share the same physical connection to a SpaceWire device, and receive only the packets they are interested in.

Applications or services may also wish to receive packets based on their protocol ID, which should be the second byte in an appropriately formatted packet. The virtual router can also be configured to act as a protocol dispatcher, routing packets based on the second byte. This allows RMAP packets to be passed to a service which deals with RMAP requests and replies, for example, while CCSDS packets are passed to a different service.

4 BENEFITS AND LIMITATIONS

It's not always possible to exactly simulate a physical device in software, and there may be timing differences between a physical device and its virtual implementation. It may also take some time to write a virtual device, although the STAR-Dundee API is designed to simplify this development and provides a number of re-usable components for this purpose.

As long as any testing is also performed on physical hardware at some stage, the use of virtual devices can provide huge benefits during initial testing and integration, for example.

Virtual devices can be used to prototype a device or new features in an existing device. They can be used to replace a device which is not available or has a fault, or to try out a new device being considered for use. Virtual networks allow different network architectures to be tested without the time consuming task of connecting devices and SpaceWire cables.

The additional debugging information that can be provided in a virtual device could be invaluable in identifying problems. Similarly the monitoring and analysis that can be performed on a virtual link can provide information which might not be obtainable in a physical network.

5 SUMMARY

For equipment developers virtual devices provide a simple method of prototyping features before implementing these features in hardware. For equipment suppliers virtual devices can allow users to test virtual implementations of hardware prior to purchase, or to begin development while waiting on delivery. For network designers

virtual networks can be constructed to rapidly prototype and test potential network architectures, without needing to connect SpaceWire cables to devices.

The ability to integrate devices and subsystems into a network without requiring the physical hardware to be present means that devices or subsystems developed at separate geographical locations can be virtually integrated at an earlier stage of development. With the additional debugging and analysis capabilities available in a virtual network, potential problems in a system can be discovered at this stage, rather than towards the end of the project when the components are physically integrated and correcting errors is more costly.

6 REFERENCES

1. STAR-Dundee, http://www.star-dundee.com, STAR-Dundee Website.

2. S. Mills, C. McClements, S.M. Parkes, "STAR-Launch and Network Discovery", International SpaceWire Conference 2010, St Petersburg, Russia, 22–24 June 2010.

3. STAR-Dundee, http://star-dundee.com/products.php, STAR-Dundee SpaceWire Products, STAR-Dundee Website.

4. S. M. Parkes, "Virtual Satellite Integration", DASIA (Data Systems In Aerospace) 2001, Nice, France, 28 May – 1 June 2001.

5. S. Mills, S.M. Parkes, R. Vitulli, "The SpaceWire Internet Tunnel and the Advantages it Provides for Spacecraft Integration", International SpaceWire Conference 2008, Nara, Japan, 4–6 November 2008.

6. ECSS, "SpaceWire, Links, Nodes, Routers and Networks", Standard ECSS-E-ST-50-12C, Issue 2, European Cooperation for Space Standardization, July 2008.

Wednesday 23 June

Components 1

SPACEWIRE RMAP IP CORE

Session: SpaceWire Components

Long Paper

Chris McClements

STAR-Dundee, c/o School of Computing, University of Dundee, Dundee, Scotland, UK

Steve Parkes

University of Dundee, School of Computing, Dundee, Scotland, UK

E-mail: chris@star-dundee.com, sparkes@computing.dundee.ac.uk

ABSTRACT

The SpaceWire RMAP core is a VHDL IP core implementing the Remote Memory Access Protocol (RMAP). RMAP has been defined by the SpaceWire Working Group and standardised by the European Cooperation for Space Standarization (ECSS) as standard number ECCS-E-ST-50-52C. RMAP allows devices to read and write from memory spaces in a standard way, increasing device interoperability and reducing development time. The SpaceWire RMAP IP core provides a well tested, easy to use core for systems that need RMAP capability.

1 INTRODUCTION

The RMAP IP core was developed by the University of Dundee under contract from the European Space Agency (ESA). It is available from ESA for use on European space missions or projects and available from STAR-Dundee for other applications. The core is highly configurable and can be used as an RMAP target or initiator. The core can be implemented in a number of technologies, including various radiation tolerant FPGAs.

2 BACKGROUND

The Remote Memory Access Protocol (RMAP) standard is now available from the ECSS website as ECCS-E-ST-50-52C [1]. The RMAP standard was written by the University of Dundee with support from members of the SpaceWire Working Group. RMAP is a SpaceWire protocol that provides a standard mechanism for reading from, and writing, to memory in a remote SpaceWire node. The RMAP protocol has already been designed into the ESA SpW-10X router ASIC [2] and into missions like Bepi-Colombo [3][4], MMS [5], and ExoMars rover [6].

3 RMAP IP CORE OVERVIEW

3.1 FUNCTION

There are two main functions which can be selected by configuration at synthesis time. The first type is referred to as the “initiator” RMAP Interface which sends out RMAP commands and receives replies. The second type is the “target” RMAP Interface which receives RMAP commands, executes them and sends out any required replies. The RMAP core has a wrapper which connects to the ESA/Dundee SpaceWire-b core [7] [8] which handles the SpaceWire point-to-point link protocol. The RMAP core target and initiator functions are illustrated in Figure 1. The core is shown in three configurations; target only, initiator only or both.

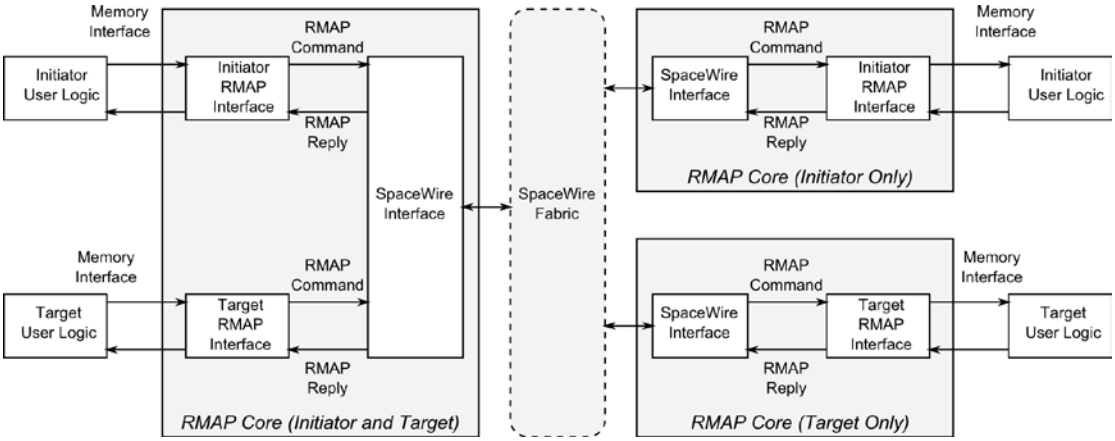


Figure 1: RMAP IP core Data Flow

RMAP commands are initiated in the initiator user logic, encoded as RMAP packets in the “Initiator RMAP Interface”, sent over the SpaceWire link as an RMAP packet, decoded by the “Target RMAP Interface” and data or information is passed to the target user logic after authorisation of the command. The “Target RMAP Interface” formats an RMAP reply packet which is sent over the SpaceWire interface, decoded by the “Initiator RMAP Interface” and the reply data/status information is passed to the initiator user logic.

3.2 ARCHITECTURE

The architecture of the RMAP core is illustrated in Figure 2. The SpaceWire CODEC implements the SpaceWire serial point to point protocol, ECSS-E-ST-50-12C [9], and provides FIFO ports to the Protocol Input and Output blocks. The Protocol Input and Output blocks determine the destination of packets dependent on the packet header. The protocol handlers can bypass data from the RMAP core if the protocol identifier does not identify the packet as an RMAP packet.

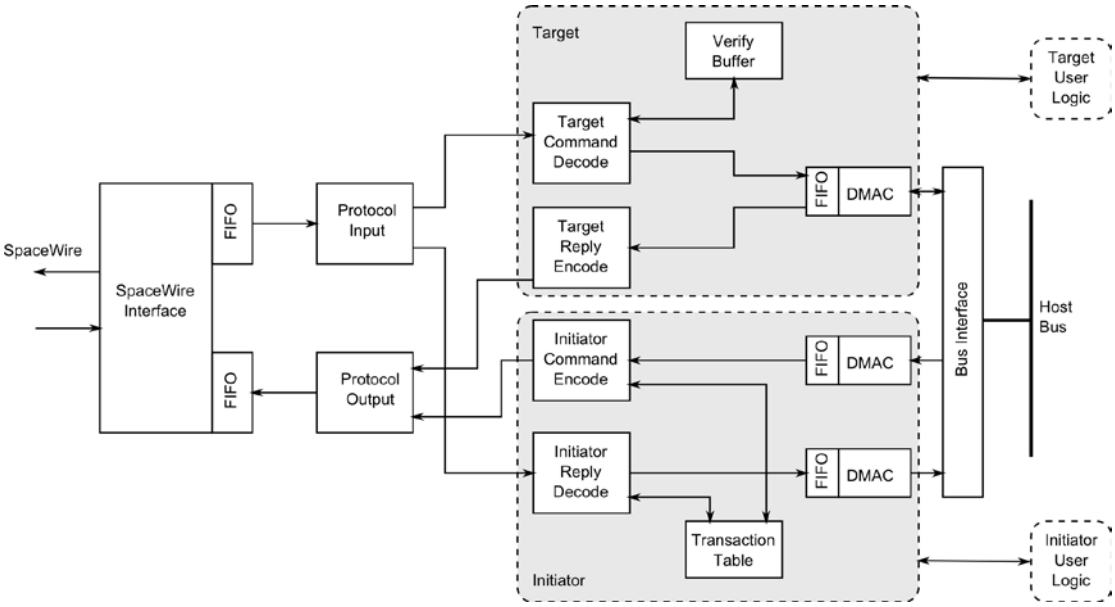


Figure 2: RMAP IP core Architecture

The Target RMAP Interface decodes RMAP command packets, reads or writes data from the host bus and returns RMAP reply packets. If the RMAP command is a verified write command the target writes the command data to the verify buffer before it is transferred to host memory. The Initiator RMAP Interface accepts commands into the initiator transaction table, encodes RMAP command packets, decodes reply packets and outputs status information. The target and initiator interact with the user memory space using DMA controllers.

3.3 MEMORY INTERFACE

The RMAP controller interface to memory is modelled on the AMBA AHB bus standard which provides a pipelined control/data bus transfer model. Data is transferred to and from the bus in bursts using internal burst FIFOs in the RMAP core. The bus can be configured for different bus size widths, byte order and bit swapping operations.

3.4 CONFIGURATION

The RMAP core is configured using VHDL generics. The configuration options of the core include the ability to implement target only logic, initiator only logic, or both target and initiator; configuration of the host bus width, the burst transfer depth and the byte/bit ordering of the RMAP packet data; watchdog timer on bus transfers; maximum number of outstanding initiator commands and therefore the initiator transaction table size; configuration of the internal FIFO sizes and the target verify buffer size.

4 USING THE RMAP CORE

4.1 TARGET

The target command logic is responsible for decoding RMAP command packets and executing the specified command, e.g. write. RMAP command headers are checked

for validity and the set of RMAP command authorisation parameters are passed to the host for authorisation. The host can check the memory address, command and other parameters, and decide to authorise or discard the command. When the RMAP command is a write command, and authorisation has been given by the host, data is placed in user memory by the target DMA controller.

The target reply logic is responsible for sending RMAP reply packets with the status of commands and additional data when a read command is performed. The status is dependent on the validity of the RMAP command packet and the authorisation response of the host. Reply data is read from the host user memory by the DMA controller and sent in the RMAP reply packet.

4.2 INITIATOR

The RMAP Initiator Handler uses several memory structures inside the RMAP core and inside initiator user memory. The structures are used to control the passing of commands from initiator user memory to the RMAP core and the passing of replies from the RMAP core to initiator user memory. The RMAP initiator data structures and data flow is depicted in Figure 3.

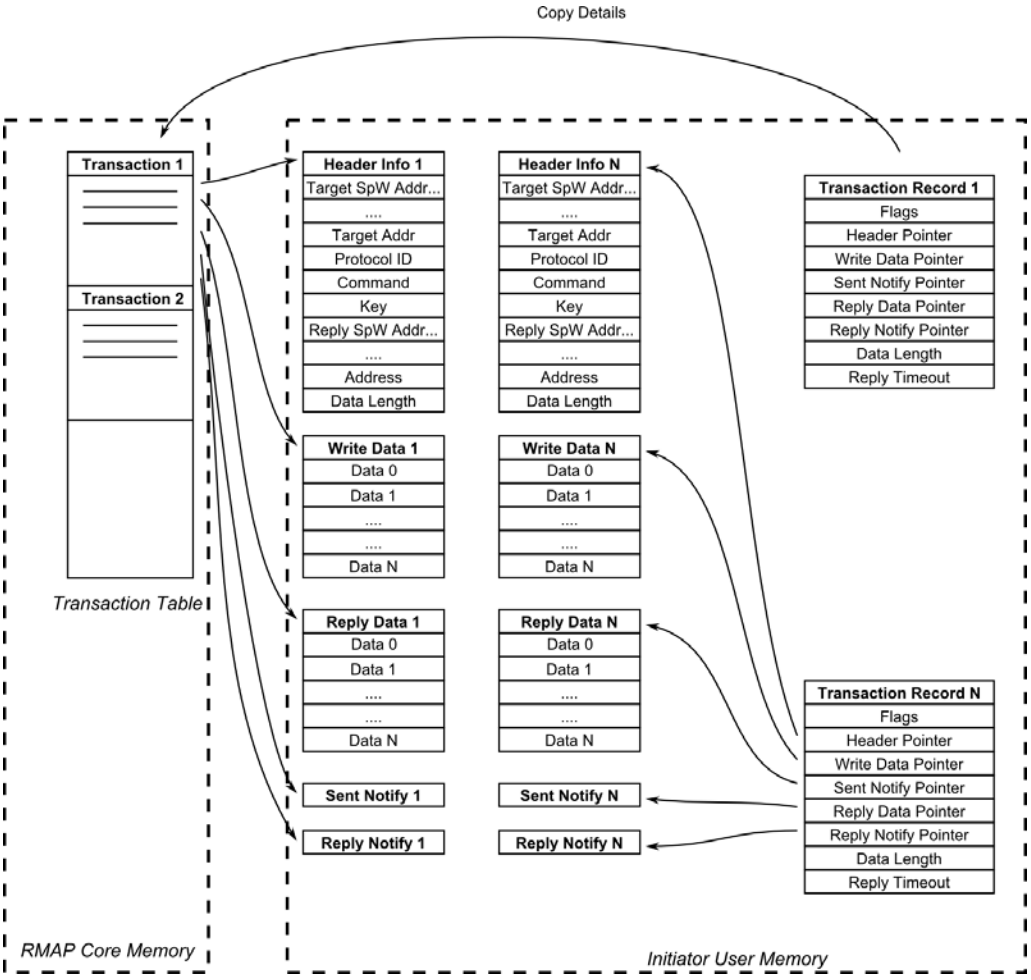


Figure 3: Initiator Data Structures

In the initiator user memory there are four possible memory areas or buffers associated with each RMAP command: transaction details record, header information, write data, and reply data.

The transaction details array holds the following information: a pointer to the command header in user memory, a pointer to any data to be sent with a write or read-modify-write command, a pointer to the memory location for sent notification, a pointer to space for a reply to a read or read-modify-write command, a pointer to the memory location for reply notification, the length of data to be read or written and the reply time-out value.

The header information buffer holds the RMAP command header information including the target SpaceWire address and the reply address. The write data buffer holds any data to be sent with a write or read-modify-write command. The reply data buffer is reserved space into which any data associated with a read or read-modify-write command will be written. The length of the buffer is given by the data length field and will not be overwritten by the core, even when a read reply packet is received with more data than is in the buffer or the RMAP header record data length field is greater than the transaction record data length.

4.3 SENDING A COMMAND

To send an RMAP command the host sets up the header of the command in a header information buffer, any data to be sent with the command in a write data buffer and space for any reply in a reply data buffer. The user then creates a transaction record with pointers to the header information buffer, write data buffer and reply data buffer along with information about the amount of data in these buffers. It also provides pointers to memory locations (or registers) where sent and reply notifications are to be made. Finally it adds into the transaction record a reply time-out value which is set in micro seconds or can also be infinite. Once the transaction record is complete the initiator user application informs the RMAP core that is has an RMAP command to send and passes the RMAP core a pointer to the corresponding transaction record.

If the transaction details record flags field indicates that the command is expecting a reply the command is not started (sent) until there is room for another transaction in its outstanding transaction array. The RMAP core will then send the command by copying the header information from user memory to the SpaceWire interface, adding any detail necessary (e.g. header CRC). The header information checked for errors before sending begins. Any errors which are detected in the header are recorded and output on the status interface and to the notify sent register, if used.

If there is any write data to be sent this will be copied from the write data buffer in user memory to the SpaceWire interface and appending the data CRC. Finally an EOP marker will be added to complete the packet. The initiator user application will be informed that the command has been sent by the RMAP core writing the transaction ID and status to the memory location specified by the sent notify pointer in the transaction details array element.

4.4 RECEIVING A REPLY

When an RMAP reply is received the core searches the outstanding transaction array for an entry with a transaction identifier that matches the transaction identifier of the reply. Assuming there is a match the core then writes any data from a read or read-modify-write reply to the user memory location specified by the reply data pointer for the corresponding entry in the transaction details array. The RMAP core writes the transaction identifier and status to the memory location specified by the reply notification pointer in the transaction details array entry. When this has been done the relevant entry in the outstanding transaction array is cleared freeing it for use by another RMAP transaction. The core can generate transaction Ids automatically to avoid the change that the user logic may use duplicate identifiers.

4.5 TRANSACTION DETAILS RECORD

The transaction details record is initialised in user memory by the host application when it wishes to send an RMAP command. The format of the transaction details record is illustrated in Table 1. The flags field is a bit mask which holds properties of the transaction record such as notify on send, wait forever, etc.

	31 23 15 7 0
0	Unused (31:16) | Flags (15:0)
1	Header Pointer (31:0)
2	Write Data Pointer (31:0)
3	Sent Notify Pointer (31:0)
4	Reply Data Pointer (31:0)
5	Reply Notify Pointer (31:0)
6	Unused (31:24) | Data Length (23:0)
7	Reply Timeout (31:0)

Table 1: Transaction Details Record Memory Setup

4.6 HEADER INFORMATION RECORD

The header information record holds information on the RMAP command parameters to be sent. An example header information record is stored in memory as defined by an example header in Table 2. In the example there are four target SpaceWire addresses and one block of reply SpaceWire addresses.

	31	*23*	*15*	*7* *0*
0	Target Path Address 1	Target Path Address 2	Target Path Address 3	Target Path Address 4
1	Target Address	Protocol ID	Instruction	Key
2	Reply Path Address 1	Reply Path Address 2	Reply Path Address 3	Reply Path Address 4
3	Initiator Address	Transaction ID 1	Transaction TID 0	Extended Address
4	Address 3	Address 2	Address 1	Address 0
5	Data Length 2	Data Length 1	Data Length 0	Unused

Table 2: Header Information Record Setup

5 IP VALIDATION

Verification of the RMAP core is performed using an automated VHDL test-bench which runs a series of test command scripts to check the function of the RMAP core.

The command scripts run test-cases which detect correct and incorrect behaviour of the configured RMAP core relative to the functional specification.

VHDL code coverage using the Modelsim simulator code coverage option in Modelsim SE is used to check for coverage of the complete design by the test cases. The purpose of the test cases is to show the function of the UUT is equivalent to the specification defined in the functional specification document.

6 SYNTHESIS

The RMAP core has one system clock input which clocks all flip-flops in the design except the receive clock domain of the SpaceWire link. The SpaceWire transmitter can also be clocked from a separate clock input on the core dependent on the SpaceWire link configuration settings.

A typical way to implement this RMAP core design is to run the system clock at the byte rate of the system and use a separate transmit clock to transmit the bytes at the required bit rate. For example a system which processes RMAP data at 20 Mbytes/s requires a 100 MHz transmit clock to transmit the byte data at 200 Mbps DDR, taking into account the SpaceWire data character length of 10 bits.

6.1 SYNTHESIS RESULTS

The results of synthesis runs on the Mentor Graphics Precision synthesiser are given below.

Model	AX2000			Spartan3E 1600	ProASIC3E1500
	FF	Comb	Modules	Slices	Tiles
Target Only with SpW	1425	2962	4464 (13.84%)	1134 (7.69%)	4576 (11.92%)
Initiator Only with SpW	2029	4434	6463 (20.04%)	2213 (15.00%)	7987 (20.80%)
Target and Initiator no SpW	2599	5634	8233 (26.20%)	2584 (17.52%)	10206 (26.58%)
Target and Initiator with SpW	2957	6249	9206 (29.06%)	3095 (20.97%)	11261 (29.33%)

Table 3: Area usage of RMAP core

6.2 SEU PROTECTION

It is expected that the fabric of the FPGA or ASIC technology will provide SEU protection for synchronous elements in the design (flip-flops). Typically memory blocks are not protected in silicon, therefore they should either be implemented as flip-flops, or a drop in replacement for the single and dual clocked memory blocks should be used in the final synthesised model. For example, memory blocks with error detection and correction (EDAC) using error correcting codes (ECC) are provided with the Actel Libero and designer tool chain. Critical memory blocks for SEU protection in the design are the verify buffer, transaction table and DMA controller FIFOs.

The SpaceWire interface transmit and receive FIFOs are also critical but as the RMAP protocol is used, the packet data is protected by header and data CRCs. In this case SEU protection may not be required.

7 CONCLUSION

The RMAP IP core was developed in the frame of the SpaceNet activity. This resulted in a SpaceWire interface VHDL core that includes the RMAP protocol extension to SpaceWire, which will enable users to readily implement the RMAP protocols in FPGAs or ASICs.

It is available from ESA for use on European space missions or projects and available from STAR-Dundee for other applications. The core is designed to be a highly configurable VHDL IP core which can be used as an RMAP target or initiator. The core can be implemented in a number of technologies, including the radiation tolerant Actel RTAX which is widely used in the Space industry.

8 REFERENCES

1. ECSS, “SpaceWire - Remote memory access protocol”, ECSS-E-ST-50-52C, Feb 2010, available from http://www.ecss.nl

2. S. Parkes, C. McClements, G. Kempf, S. Fishcher, P. Fabry, A. Leon, “SpaceWire Router ASIC”, International SpaceWire Conference, Dundee, 2007.

3. T. Takashima, H. Hayakawa, H. Ogawa, Y. Kasaba, M. Koyama, K. Masukawa, M. Kawasaki, S. Ishii, Y. Kuroda, BepiColombo MMO Project Data-Handling Team, “Introduction of SpaceWire Applications for the MMO Spacecraft in BepiColombo Mission”, International SpaceWire Conference, Dundee, 2007.

4. T. Yuasa, K. Nakazawa, K. Makishima, H. Odaka, M. Kokubun, T. Takashima, T. Takahashi, M. Nomachi, I. Fujishiro, F. Hodoshima, “Development of a SpW/RMAP-based Data Acquisition Framework for Scientific Detector Applications, International SpaceWire Conference, Dundee, 2007.

5. ESA, “ESA – Aurora Programme – ExoMars”, http://www.esa.int/SPECIALS/Aurora/SEM1NVZKQAD_0.html

6. Soutwest Research Institute, “MMS-SMART Home Page”, http://mms.space.swri.edu/

7. ESA, “Micro-electronics Website “, http://www.esa.int/TEC/Microelectronics/SEMLOU8L6VE_0.html”, ESA, 2010.

8. STAR-Dundee website: www.star-dundee.com

9. ECSS, “SpaceWire: Links, nodes, routers and networks”, ECSS-E-ST-50-12C, July 2008.

THEORY OF OPERATION AND V_{DD} FAULT SCENARIO FOR LVDS PHYSICAL LAYER OF SPACEWIRE

Session: SpaceWire Components

Long Paper

Jennifer Larsen

Aeroflex Colorado Springs

4350 Centennial Blvd. Colorado Springs, CO 80907

E-mail: *Jennifer.Larsen@aeroflex.com*

ABSTRACT

The SpaceWire Standard ECSS-E-ST-50-12C [3] calls for a Low Voltage Differential Signaling (LVDS) physical layer as defined in ANSI/TIA/EIA-644 [1], Electrical Characteristics of Low Voltage Differential Signaling Interface Circuits. This paper will discuss that Aeroflex Colorado Springs LVDS drivers are compatible with the ANSI/TIA/EIA-644 standard and contain a current source which generates the required voltage across a 100Ω, parallel, resistor. Laboratory results will be reported which examine a hypothetical failure mode where the supply voltage, V_{DD}, exceeds the ABSOLUTE MAXIMUM RATINGS defined in the Aeroflex Datasheet [4][5][6][9].

1 BACKGROUND

Low Voltage Differential Signaling, LVDS, is useful in applications that require low power, low noise, and high-speed point-to-point communications. The SpaceWire physical layer uses Data-Strobe (DS) Encoded LVDS to communicate serial, full-duplex, bidirectional data. Figure 1 shows a notional SpaceWire Link using the LVDS physical layer, the operation of LVDS will be explained later in this paper.

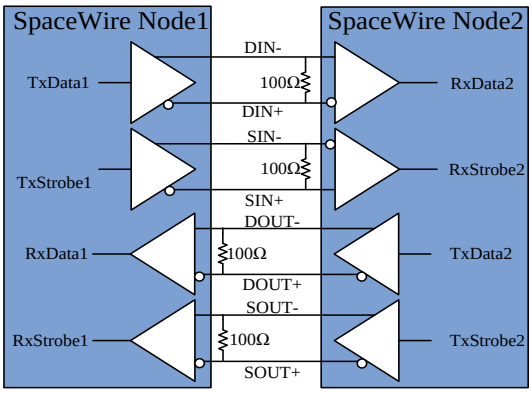


Figure 1. Single point to Point SpaceWire Link

The Input/Output signal levels are defined by ANSI/TIA/EIA-644, which is an electrical signaling standard only; it does not define a protocol. Instead, the protocol is defined in the SpaceWire Standard specification ECSS-S-ST-50-12C, which is derived from IEEE 1355-1995.

2 LVDS FUNCTIONALITY

LVDS is a method used to transmit and receive hundreds of megabits per second over differential media using a low voltage signal swing (~350mV). LVDS communications are preformed by a driver and a receiver. The driver accepts a standard Complementary Metal Oxide Semiconductor (CMOS) signal and outputs a constant current, differential, signal. The LVDS receiver senses the differential voltage across a 100Ω termination resistor and outputs a standard CMOS signal equivalent to the supply voltage. The differential aspect of LVDS allows systems to run at high data rates, with low switching power, high noise immunity, and common mode range.

The LVDS driver works by using NMOS Field Effect Transistors (FETs) to control the direction of the constant 3.5mA current source through the termination resistor. The driver current, flowing through the 100Ω termination resistor placed across the differential inputs of the receiver, generates a +/-350mV I-R drop which is sensed as a logic high/low by the receiver. The LVDS receiver has very high DC input impedance, so the majority of the driver's current flows, in a loop, from the source terminal through the 100Ω termination resistor and back into the sinking terminal.

When the driver output current direction changes, the direction of current flow across the termination resistor changes accordingly, creating the logic 1 or 0 state at the receiver output. Figure 3 shows current flow through the driver/receiver system resulting in a logic low at the receiver output. The constant current source drives +3.5mA through Q2 and into the negative half of the LVDS bus. The current reaches the termination resistor located at the receiver, flows back through the positive half of the bus, returns to the terminal drain of Q3 in the driver, and then passes into the driver VSS plane. The direction of the current flow through the termination resistor (negative to the positive), a forward voltage drop occurs and a logical low appears at the output (ROUT) of the receiver. The opposite is true for a logic high on the receiver output as shown in figure 4.

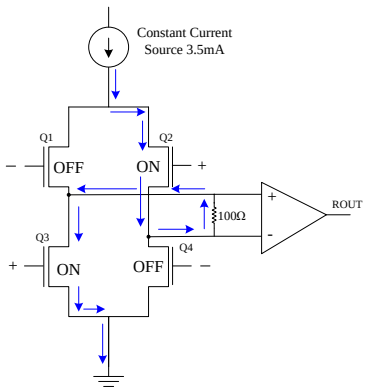


Figure 3. Logic Low (zero, 0) State

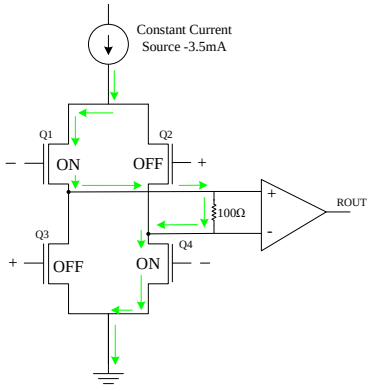


Figure 4. Logic High (one, 1) State

3 LABORATORY EXPERIMENTS OF POWER SUPPLY FAULTS [7]

UT54LVDS031LV and UT54LVDS032LV devices were used for all laboratory experiments. To demonstrate the devices response to an over voltage failure mode, experiments were performed with an 8.0V ramp on supplies and I/Os while monitoring the driver inputs (DIN), driver outputs (DOUT+/-), receiver inputs (RIN+/- across 100Ω), and receiver outputs (ROUT). All data was taken at 25°C (ambient) with either a 1 minute or 15 second dwell time at each voltage providing more than enough time for any overheating damage to present [7].

One of the experiments conducted in the lab at ambient conditions used the following test setup.

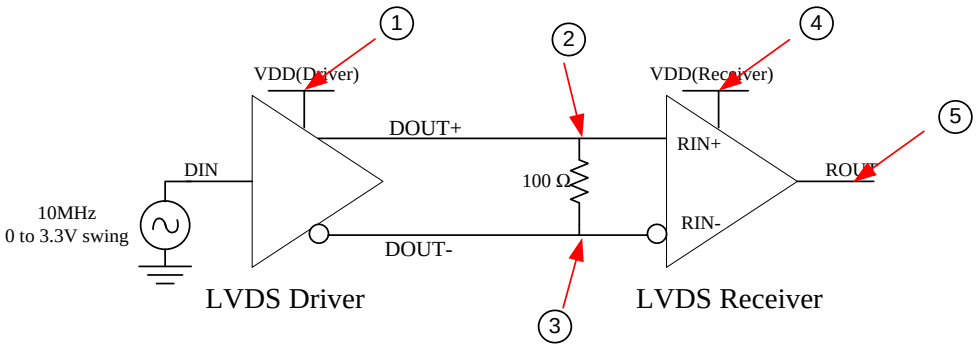


Figure 5. Example 1 Test Configuration
DIN = 10MHz with a 0 to 3.3V swing, EN = 3.3V, /EN = 0V

This test evaluated how a standard LVDS driver/receiver configuration responds to a driver power supply over voltage fault. The voltage and current of the driver and receiver power supplies was monitored. The differential output voltages on the DOUT+/RIN+ and DOUT-/RIN- signals across the 100Ω termination resistor and the output of the receiver (ROUT). VDD(Driver) was ramped from a nominal 3.3V to 8.0V. A standard lab power supply with 2.5A current limit was used for this experiment. The power supply current limiting capabilities prevented VDD on the UT54LVDS031LV driver from ramping past 8.0V.

Table 1. Example 1 Test Results

VDD(Driver)	**I VDD(Driver)**	**DOUT+**	**DOUT-**	**VDD(Receiver)**	**I VDD(Receiver)**	**ROUT**
(V) Point 1	(mA) Point 1	(V_{AVG}) Point 2	(V_{AVG}) Point 3	(V) Point 4	(mA) Point 4	(V_{AMP}) Point 5
3.3	12.49	1.30	1.41	3.3	6.66	3.12
3.6	13.17	1.33	1.44	3.3	6.63	3.04
4.0	13.73	1.35	1.46	3.3	6.62	3.20
5.0	15.95	1.47	1.56	3.3	6.56	3.02
6.0	23.31	1.83	1.92	3.3	6.54	3.20
7.0	261.0	2.51	2.61	3.3	6.07	3.12
8.0	500.0	2.38	3.20	3.3	10.50	3.42

Table 1 indicates that the standard LVDS driver/receiver configuration did not propagate a high voltage fault when VDD on the driver is ramped to 8.0V. The maximum voltage seen on the DOUT+/RIN+ and DOUT-/RIN- signals are within the absolute maximum ratings for both devices. The high voltage stress on the driver

VDD permanently damaged the driver device without propagating the fault to the receiver. The UT54LVDS032LV receiver continued to function after the fault condition.

The next experiment conducted in the lab at ambient conditions used the following test setup.

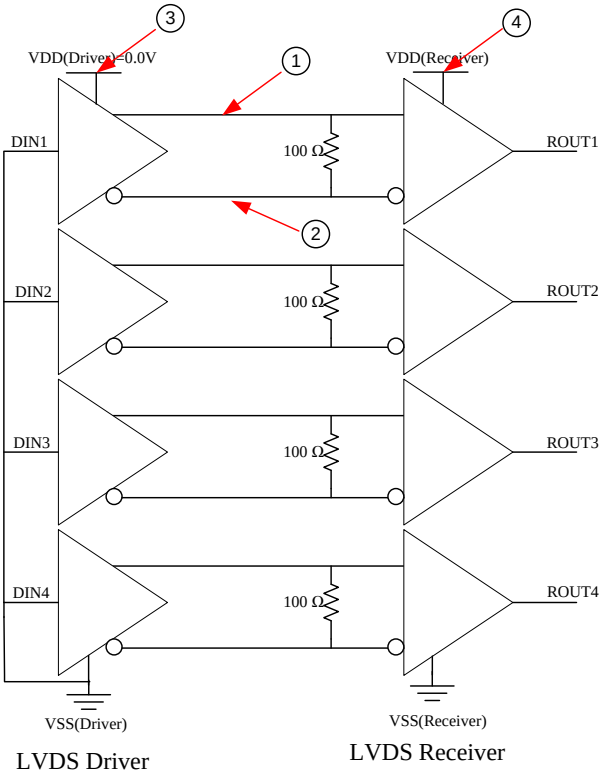


Figure 6. Example 2 Test Configuration
EN = /EN = 0.0V

This test evaluated the differential lines (DOUT+/RIN+ and DOUT-/RIN-) with the receiver powered and the driver unpowered (cold spare mode). The inputs of the driver (DIN) were set to 0.0V and the supply voltage of the receiver, VDD(Receiver), was ramped to 8.0V. The voltage and current of the receiver and driver power supplies and the +/- differential signals were monitored.

Table 2. Example 2 Test Results

VDD(Receiver)	**I VDD(Receiver)**	**VDD (Driver)**	**I VDD(Driver)**	**DOUT1+**	**DOUT1-**
(V) Point 4	(mA) Point 4	(V) Point 3	(mA) Point 3	(V_{AVG}) Point 1	(V_{AVG}) Point 2
3.3	9.54	0.0	0.07	1.23	1.45
3.6	10.11	0.0	0.11	1.93	1.93
4.0	10.87	0.0	.10	2.72	2.73
5.0	15.71	0.0	0.08	4.83	4.81
6.0	15.71	0.0	0.08	5.83	5.82
7.0	17.86	0.0	0.11	6.80	6.77
8.0	19.70	0.0	0.11	0.20	0.24

Table 2 indicates this standard LVDS driver/receiver configuration does propagate receiver supply over voltages. Although over voltages can propagate, faults are not propagated based on further analysis. Additionally, at 8V the voltage on the LVDS inputs drops significantly indicating the receivers experience catastrophic failure but the voltage out of the LVDS inputs remains within the recommended operating range when VDD exceeds approximately 8V.

Another example experiment conducted in the lab at ambient conditions used the following test setup.

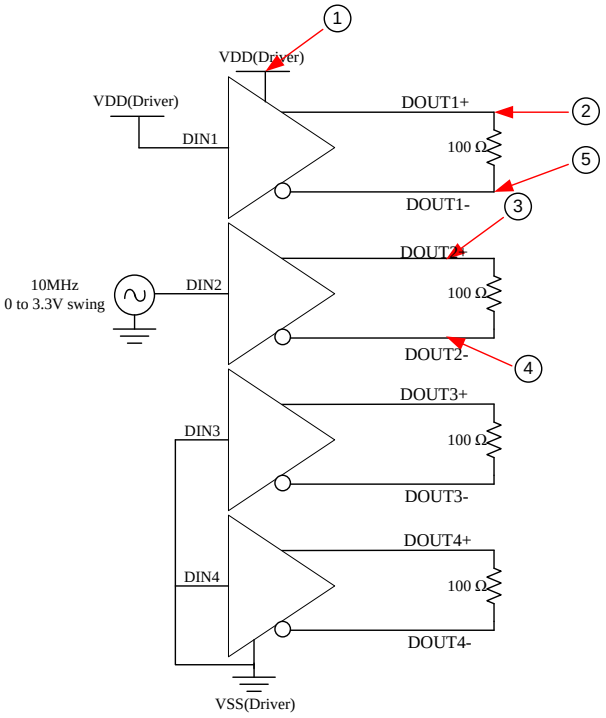


Figure 7. Example 3 Test Configuration
EN = 3.3V, /EN = 0V
DIN1 = VDD(Driver) DIN2 = 10MHz with a 0 to 3.3V swing DIN3 = DIN4 = VSS = 0.0V

This test evaluated the differential line (DOUT+ and DOUT-) performance when VDD of the driver was ramped from 0.0V to 8.0V. This was accomplished by setting the enable signals on the UT54LVDS031LV to EN=3.3V and /EN=0.0V. Input #1 (DIN1) was set to VDD(Driver), input #2 (DIN2) was stimulated with a 10MHz square wave with 0.0 to 3.3V peak-to-peak, and inputs #3 and #4 were set to 0.0V. The voltage and current of the driver power supply, voltage across the termination resistor across DOUT1, voltage at DOUT1+, voltage at DOUT2+, and voltage at DOUT2- were monitored.

Table 3. Example 3 Test Results

VDD(Driver)	I VDD(Driver)	DOUT1+	DOUT2-	DOUT2+	DOUT1+/-
(V) Point 1	(mA) Point 1	(V_{AVG}) Point 2	(V_{AVG}) Point 3	(V_{AVG}) Point 2	(V_{AVG}) Point 2
3.3	16.48	1.36	1.10	1.40	0.34
3.6	17.07	1.42	1.10	1.36	0.35
4.0	17.91	1.45	1.41	1.07	0.37
5.0	24.61	1.61	1.29	1.53	0.42
6.0	86.08	2.12	1.82	1.82	0.62
7.0	169.31	3.64	3.18	2.16	1.16
8.0	758.81	0.54	0.58	0.60	0.00

Table 2 indicates the UT54LVDS031LV does not propagate supply over voltages when the voltage on the LVDS input pin is within the recommended range (DOUT2) but does propagate over voltages when the LVDS input pin follows the supply overvoltage (DOUT1). The voltage across DOUT1's resistor scales proportionally with the overvoltage, showing increasing current due to increasing voltage.

Although over voltages can propagate, faults are not propagated as the current sourced by DOUT1+ is returned through DOUT1-. Insufficient current would flow into a receiver to cause damage or degradation per section 4.0. Additionally, at 8V the voltage on the LVDS lines drops significantly indicating the drivers experience catastrophic failure but the outputs fail to a voltage within the recommended operating range for VDD ≥ 8V.

The last example provided for this paper was conducted in the lab at ambient conditions used the test setup shown in figure 8.

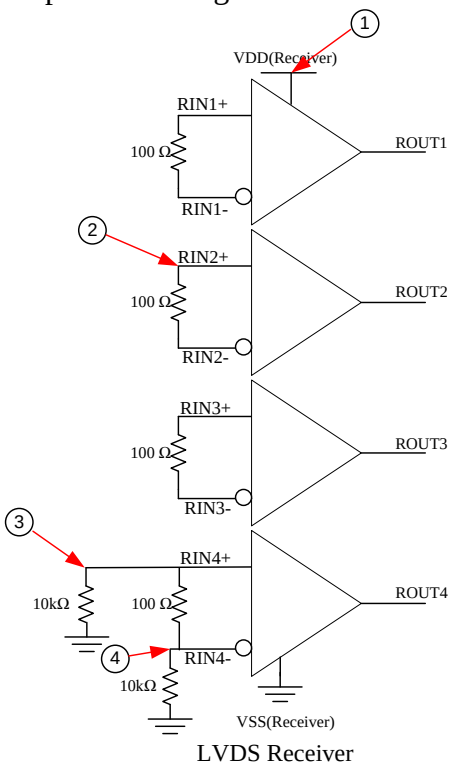


Figure 8. Example 4 Test Configuration
EN = /EN = 0V

This test evaluated the receiver differential inputs (RIN+/RIN-) performance with some inputs terminated with 100Ω resistors and one pair of inputs pulled to 0.0V (VSS(Receiver)) using 10kΩ resistors, and VDD(Receiver) was ramped from 0.0V to 8.0V. The voltage and current of the receiver power supply, and selected RIN+/RIN- inputs monitored. A standard lab power supply with 2.5A current limit was used for this experiment. Table 4 indicates the UT54LVDS032LV does propagate receiver supply over voltages out of its RIN inputs.

Table 4. Example 4 Test Results

VDD(Receiver)	**I VDD(Receiver)**	**RIN2+**	**RIN3+**	**RIN3-**
(V) Point 1	(mA) Point 1	(V_{AVG}) Point 2	(V_{AVG}) Point 3	(V_{AVG}) Point 4
3.3	7.93	2.36	0.01	0.01
3.6	7.98	3.33	0.00	0.00
4.0	8.68	3.87	0.03	0.03
5.0	13.61	4.90	0.13	0.11
6.0	33.43	5.87	0.25	0.25
7.0	84.81	6.85	0.48	0.47
8.0	1188.28	0.25	0.27	0.21

Although over voltages can propagate, faults are not propagated as there is insufficient current sourced out of the RIN pins, calculated from the voltage drop across the 10 kΩ pull-down resistors, to damage or degrade the LVDS outputs per section 4.0. Additionally, at 8V the voltage on the LVDS inputs drops significantly indicating the receivers experience catastrophic failure but the voltage out of the LVDS inputs remains within the recommended operating range when VDD exceeds approximately 8V.

This experiment shows that pull-down resistors could be used to preclude receiver supply over voltages from propagating to a driver. However, these are not necessary as there is insufficient current to propagate a fault. Also, the LVDS lines would only show the receiver supply overvoltage if the driver was tristated or powered-off, as an active driver would overpower the receiver supply over voltages and hold the LVDS lines at operating levels.

4 LVDS I/O Current Reliability Analysis

The metal current density design rules for the 0.25µm LVDS technology allow a DC current of 5.98mA on the LVDS I/O while remaining within allowed specifications**.** After reviewing the reliability assessment the UT54LVDS031LV, UT54LVDS032LV, UT200SpWPHY01, and the UT200SpW4RTR these devices can tolerate 100µA of current per pin indefinitely without damage or degradation for a 15 year mission.

Additionally, from Table 4, an LVDS receiver experiencing overvoltage could back-drive up to 57µA per LVDS line to an LVDS driver. If all four pairs of LVDS lines were connected, a total of approximately 460µA could be back-driven. 460µA is more than an order of magnitude below the 5.98mA calculated threshold. Therefore,

no faults would be propagated from an LVDS receiver experiencing over voltages to an LVDS driver.

5 FAULT CONDITIONS AND RESULTS

The fault conditions examined focus on demonstrating the response of Aeroflex LVDS drivers and receivers to voltage stress above the ABSOLUTE MAXIMUM RATING for VDD and the I/O. The tests were preformed with the UT54LVDS031LV and UT54LVDS032LV devices at room temperature (25°C).

The tests described above show that over voltages can be seen on the LVDS I/O, but that faults are not propagated because the current sourced over the LVDS lines by the part experiencing the overvoltage is not sufficient to cause damage or degradation to the other LVDS part. The experiments also show that sufficiently high voltage applied to the device resulted in transistor break down causing permanent damage with no overvoltage propagation.

The Aeroflex LVDS drivers are current mode outputs with a constant current source capable of driving a 3.5mA nominal current. The results discussed are not guaranteed by Aeroflex. Any operation outside of the ABSOLUTE MAXIMUM RATINGS, as stated in the datasheet and/or SMD may affect device reliability and performance.

6 REFERENCES

[1] Telecommunications industry Association, “Electrical Characteristics of Low Voltage Differential Signaling (LVDS) Interface Circuits ANSI/TIA/EIA-644”, January 30, 2001.

[2] IEEE P1355, “Standard for Heterogeneous InterConnect (HIC) IEEE 1355-1995”, Conference Title, Location, June 12, 1996.

[3] ESA Publications Division, “SpaceWire Standard Document ECSS-E-ST-50-12C”, The Netherlands, July 31, 2008.

[4] Aeroflex, “UT54LVDS031LV 3.3-VOLT QUAD DRIVER Datasheet”, Colorado Springs, Colorado, February 2006.

[5] Aeroflex Colorado Springs, “UT54LVDS032LV 3.3-VOLT QUAD RECEIVER Datasheet”, Colorado Springs, Colorado, March 2006.

[6] Aeroflex Colorado Springs, “UT200SpWPHY01 SpaceWire Physical Layer Transceiver Datasheet”, Colorado Springs, Colorado, February 2008.

[7] Barry Cook, “SPACEWIRE PHYSICAL LAYER FAULT ISOLATION” Session: Test, SpaceWire Components Short Paper, 4Links Limited, Bletchley Park, MK3 6EB, England

[8] Aeroflex Colorado Springs, "Theory of Operation and VDD Fault Scenario Application Note", Colorado Springs, Colorado, March 2010

[9] Aeroflex Colorado Springs, "UT200SpW4RTR 4-Port SpaceWire Router Datasheet", Colorado Springs, Colorado, February 2010.

SPACE-QUALIFIED TRANSCEIVER FOR SINGLE-LINK SPACE WIRE INTERCONNECT

Session: SpaceWire Components

Long Paper

Vladimir Bratov, Vladimir Katzman, Glenn P. Rakow

Vladimir Bratov and Vladimir Katzman are with Advanced Science and Novel Technology (ADSANTEC) 27 Via Porto Grande, Rancho Palos Verdes, CA 90230, USA.

Glenn P. Rakow is with NASA Goddard Space Flight Center, Maryland, USA.

E-mails: vbratov@adsantec.net, vkatzman@adsantec.net, glenn.p.rakow@nasa.gov

ABSTRACT

A new interconnect technique utilizing three-level pulses for transmission of data and clock information is presented. This technique is fully compatible with the SW protocol but utilizes only one differential data channel. The second channel normally required for the SW data/strobe link now operates as a redundant line controlled by special circuitry that monitors the integrity of both channels. A transceiver chip that implements a version of the developed technique has been designed and fabricated in a 180*nm* BiCMOS technology suitable for the development of space electronics. Following the successful tests of the transceiver, a SW port with three-level interface has been designed.

1 INTRODUCTION

Performance improvement that is required for the success of future space missions dictates the migration from low-speed parallel to high-speed serial data interconnect interfaces. In contrast with ground-based electronics, achievement of high data transmission rates in space-oriented interconnect systems presents a significant challenge due to tight requirements for their stability in harsh operational conditions. Application of the existing techniques for stability improvement, such as the ones described in [1], inevitably degrades the achievable speed-power performance.

Space plug-and-play avionics is an emerging technology that can alleviate serial data interconnect shortcomings in present-day solutions. It is based on the switching fabric active backplane architecture with robust high-speed serial interfaces. Electrical and/or optical transponders operating with Space Wire (SW) [2], optical SW (SW-Fiber), Fire Wire (FW), or Ethernet/Gigabit Ethernet protocols are required to support the associated high-speed data interconnects.

Unfortunately, the achievable performance of the copper-based SW interconnects is limited by the specific structure of the interface that requires two differential channels per unidirectional link to implement the data-strobe (DS) encoding scheme. Unavoidable channel-to-channel skew accompanied by signal degradation during the transmission process complicates the clock recovery process and prevents system operation at high data rates.

At the same time, the higher flexibility of the electrical SW protocol compared to its optical version makes it extremely attractive for applications in both space-oriented and ground-based systems. This paper presents a novel electrical interconnect technique based on three-level voltage pulses that facilitates the transmission of both data and clock information through one differential channel thus eliminating the channel-to-channel skew problem. As a result, operational speed above 1*Gb/s* can be achieved without modifications of the protocol. In addition, the second differential channel can be used as a redundant connection that increases the fault tolerance of the SW system. Special link integrity control algorithm and circuit have been developed to provide real-time monitoring and activation of the functional link on both transmitter and receiver sides.

The developed interconnect technique has been validated through fabrication of a test transceiver chip with three-level input-output interfaces. The chip was designed on the basis of a special library of fully-differential CML (current-mode logic) cells and functional blocks which utilize SiGe hetero-junction *n-p-n* bipolar transistors (HBT) as active components and poli-Si resistors as loading elements. Those components are available in commercial BiCMOS technologies and offer high speed and natural tolerance to harsh environmental conditions associated with space missions. Following the successful tests of the transceiver chip, a SW port with reconfigurable TL input-output interfaces operating at 1.25Gb/s data rate has been designed within the same library of cells and blocks.

The following sections describe the details of the technique (Section 2), present the design of three-level input/output blocks (Section 3) and SW transceiver chip (Section 4), and discuss the design of a SW port. (Section 5).

2 THREE-LEVEL INTERCONNECT TECHNIQUE

SW protocol relies on the synchronous transmission of two signals, one of which represents the actual data and the other one is a logic XOR function of the data and clock signals. The combination of those signals provides a possibility to reconstruct both data and clock at the receiver side in real time. The same functionality can be achieved with a different three-level (TL) interconnect technique illustrated by Fig. 1 [3].

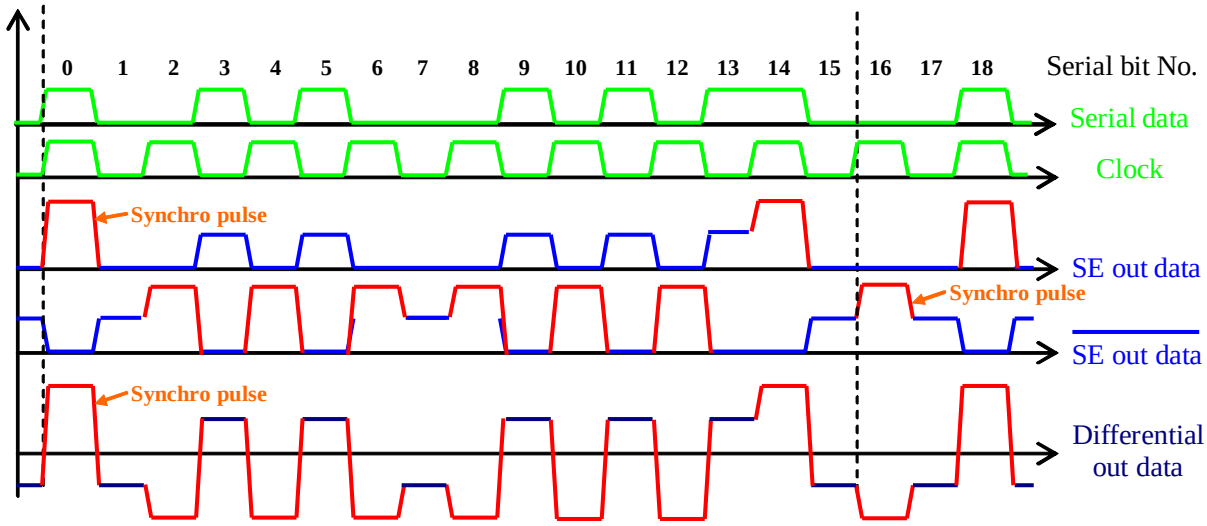


Fig. 1. Timing Diagram of Three-Level Interface.

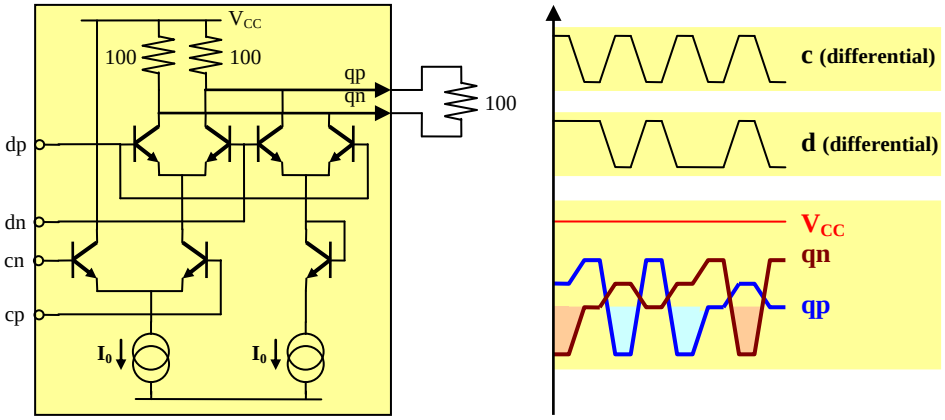
Within this technique, both data and clock information is transmitted through a single differential interconnect line. Prior to transmission, the data is processed by an output three-

level buffer which imposes synchro pulses of increased amplitude onto each odd bit in each single-ended data channel if it has a predefined logic value (either "1" or "0"). Differential signalling guarantees that the synchro pulses are always present in either the direct or the inverted output bit stream as shown in Fig. 1 for synchro pulses represented by higher "1" level. The imposed synchro pulses are retrieved from the input data stream by the receiver's three-level input buffer and converted into a half-rate clock signal required by the SW protocol.

The main difficulties of TL technique implementation are the extraction of single-ended synchro pulses and minimization of the recovered clock jitter primarily associated with the pattern-dependent data jitter.

3 THREE-LEVEL INTERFACE IMPLEMENTATION

The synchro-pulses in TL interface should be represented either by higher levels of logic "1" state or lower levels of logic "0" state. At the same time, the differential bit stream must be compatible with the LVDS (low-voltage differential signalling) electrical interface [4] as defined by the SW standard [2]. To overcome this difficulty, a pseudo-LVDS TL output buffer has been designed. The buffer incorporates a complex logic function of clock "c" and data "d" signals as shown in Fig. 1a. During the periods of negative input clock signal, it generates a logic "1" level equal to "Vcc"-0.4*V* and a logic "0" level equal to "Vcc"-0.8*V* on the external 100*Ohm* load. During the periods of positive input clock signal, the levels are changed to "Vcc"-0.6*V* an "Vcc"-1.2*V* respectively. The resulting signals are shown in Fig. 1b, where filled regions represent the overhead synchro pulses.



a. b.

Fig. 2. TL Output Buffer (a) and Its Signals (b).

The input TL buffer needs to detect the single-ended overhead pulses and convert them into a half-rate clock signal, as well as to process the differential data signals as normal LVDS signals. To perform these functions, the block includes a universal input buffer with high tolerance to common-mode voltage variation [5] that processes the data signals, and a dual comparator that includes two individual comparators connected in series as shown in Fig. 3. The first comparator compares both direct "dp" and inverted "dn" data streams with a threshold voltage "V_{CM}" derived from the input minimum voltage level detected by a peak detector. The resulting signals "q1p" and "q1n" are combined by the logic OR function and compared by the second comparator to the threshold voltage "V_{TH}" derived from the internal

CML logic levels. This dual-threshold technique is self-adjustable to the input common-mode voltage variation within the limits defined by the CML buffer and provides a reliable clock reconstruction within the temperature range from -25°C to 125°C, ±5% of power supply variation, and ±3σ process parameters variation.

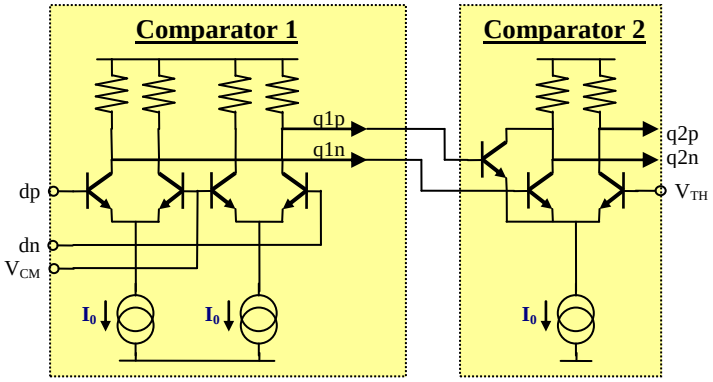


Fig. 3. Dual Comparator.

Computer simulations of the TL interface with realistic package equivalents have proved the acceptable quality of the detected clock signal. The worst-case jitter of the signal does not exceed 0.05UI at the frequency of 1.25*GHz*.

4 TRANSCEIVER TEST CHIP

The discussed TL input/output blocks were incorporated into a complex transceiver chip designed in a commercial 180*nm* SiGe BiCMOS technology. The chip consists of a number of blocks including a multiplexer with TL output interface, a demultiplexer with TL input interface that converts the input data into 40-bit wide parallel output words, and a phase-locked loop (PLL) that uses the reconstructed clock as a reference signal. A microphotograph of the chip is shown in Fig. 4.

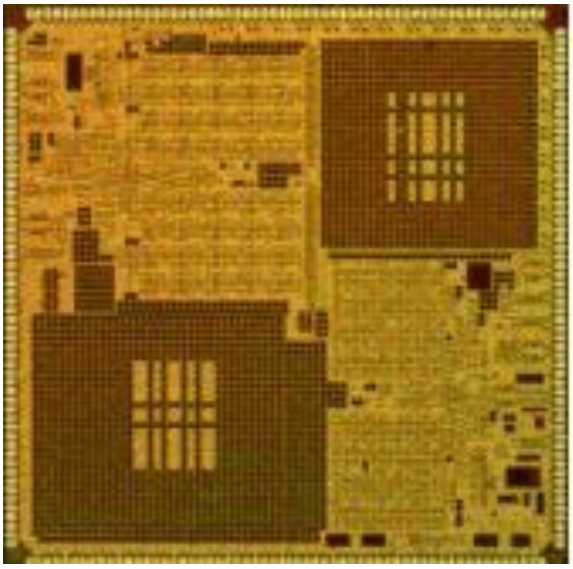
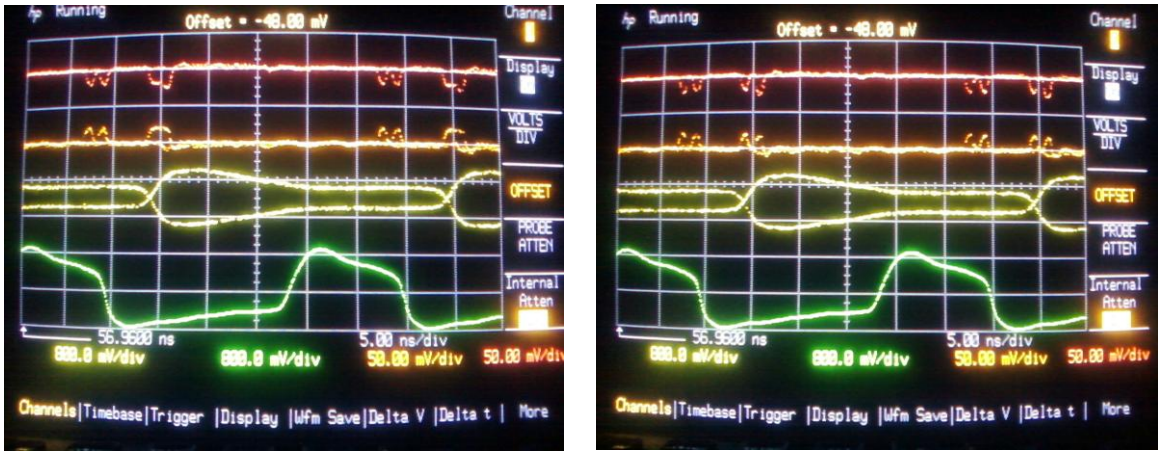


Fig. 4. Microphotograph of the Transceiver Chip.

In this version of TL interface, the overhead pulses are imposed onto each 40-th bit of the data bit stream. The fabricated chip was packaged into a 176-pin TQFN package and tested in a loop mode when the multiplexer's output signal is externally applied to the inputs of the

demultiplexer. The results presented in Fig. 5 demonstrate the transceiver operation in case of equal and opposite data bits overlapping the synchro pulses. These bits are toggling in each 40-bit word. The oscilloscope is triggered by a divided-by-40 clock signal, which causes the double lines on the screen.



a. b.

Fig. 5. Oscilloscope Screen Shots with Equal (a) and Opposite (b) Bits Corresponding to Synchro Pulses.

The top two lines (red and orange) show the direct and inverted TL signals at the multiplexer outputs. The yellow lines show the demultiplexer output signal corresponding to one of multiplexer's toggling input signals. The bottom line (green) shows the synchronized divided-by-40 clock from the demultiplexer's PLL in a lock state.

5 SW PORT DESIGN

Successful test results of the transceiver chip have provided the basis for the low-risk design of the SW port that may serve as a part of a SW switching fabric or other SW devices. The port is designed within the same CML library of cells and blocks. The main parts of the port include a reconfigurable TL transmitter (RCT) with a line integrity control circuitry (LIC) and a reconfigurable TL receiver (RCR) with the same circuitry. The port operates at maximum data rates up to 1.25*Gb/s*. Its block diagram is shown in Fig. 6.

5.1 RECONFIGURABLE TRANSMITTER AND RECEIVER

RCT operates either as a standard SW device where it transmits data and strobe signals through two differential output lines, or as a TL device where it imposes one-bit long synchro pulse on top of each odd bit in both direct and inverted data stream if the corresponding bit has the logic "0" value. The type of the operational mode is defined by control signal "ms2".

Distribution of the output signals among output channels is defined by control signal "ms1T". In the normal SW mode, this switch defines the assignment of data and strobe signals to two output channels. In the TL mode, it works in combination with the "line hardware error" signals "lheT" and defines the active TL channel. Those error signals indicate faulty states of output differential line and are generated by LIC block of RCT that monitors the output lines for short or open conditions.

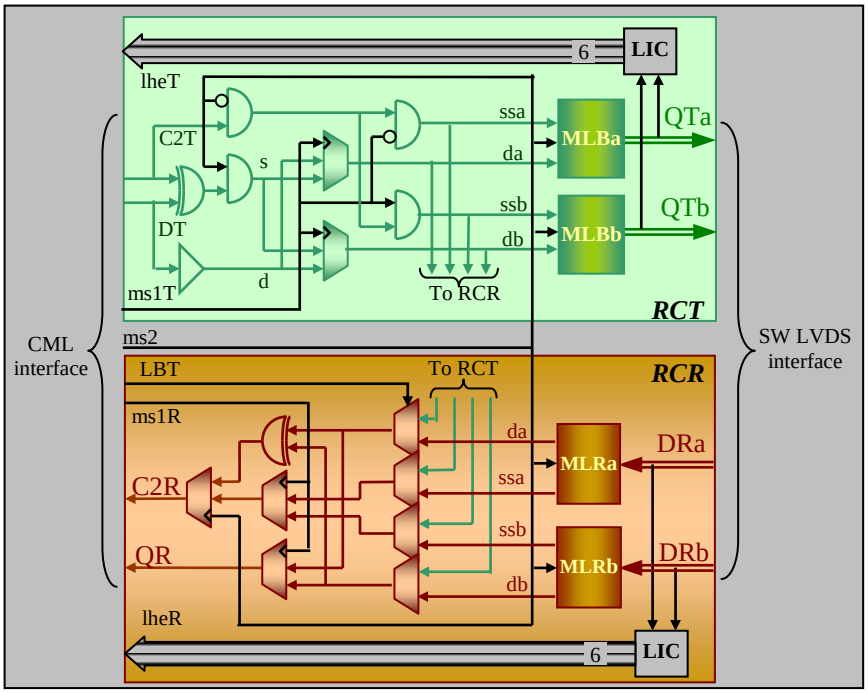


Fig. 6. Block Diagram of SW TL Routing Port.

RCR can operate in the same two modes depending on the state of "ms2" signal. In the normal SW mode, RCR reconstructs the incoming data and half-rate clock signals using the standard XOR logic function. In the TL mode, it extracts the same half-rate clock from the overhead synchro pulses as described in Section 3. RCR also includes the LIC block identical to that of RCT. The "ms1R" signal is used for activation of a certain input channel or identification of data and strobe signals. The optional "loop-back test" mode may be activated by the "LBT" signal. In this case, the transmitter outputs are sent back to the receiver bypassing the SW interface.

5.2 Line Integrity Control

The developed algorithm for the integrity control of the port's input/output lines is based on voltage logic level monitoring. The detectable signal line states of the LVDS-compatible TL interface are reproduced in Table 1, where "dp" and "dn" are the direct and inverted inputs, "qp" and "qn" are the direct and inverted outputs, $\Delta V_L = V^1 - V^0$ is the voltage logic swing, and V_H represents the positive supply voltage.

Table 1. Line State Detection Conditions.

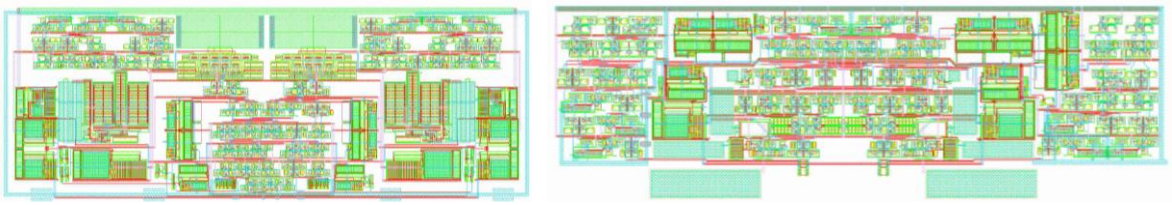
Line State	Input Interface		Output Interface	
	Logic Levels *)	Condition	Logic Levels	Condition
Normal	$V^1=V_H-\Delta V_L$ $V^0=V_H-2\Delta V_L$		$V^1=V_H-\Delta V_L$ $V^0=V_H-2\Delta V_L$	
Broken	$V^1=V_H$ $V^0=V_H-3\Delta V_L$	$V_{dp}= V_{dn}$	$V^1=V_H$ $V^0=V_H-3\Delta V_L$	$V^1=V_H$
Shorted together	$V^1=V_H-1.5\Delta V_L$ $V^0=V_H-1.5\Delta V_L$	$V_{dp}= V_{dn}$	$V^1=V_H-1.5\Delta V_L$ $V^0=V_H-1.5\Delta V_L$	$V_{qp}= V_{qn}$
Shorted to GND	$V^1=V_H/2$ $V^0=0$	$V_{dp}<V_H/2$ $V_{dn}<V_H/2$	$V^1=V_H/2$ $V^0=0$	$V_{qp}<V_H/2$ $V_{qn}<V_H/2$

*) – transmitter levels may be shifted

Looking at Table 1, it is obvious that three independent line controllers are needed at both the transmitter and receiver sides. These circuits are implemented as voltage comparators with threshold voltages $V^{R1}=V_H-\Delta V_L/2-\Delta V_{EF}$, $V^{R2}=(V_H+V_L)/2$, and $V^{R3}=Vcc/2$. They generate a combined error signal corresponding to any of the line faulty condition. This signal can be used by external processor for the activation of the operational interconnect line.

5.3 LAYOUT DESIGN

The layout of the port has been designed using standard cells from the CML library. The top view of RCT and RCR layouts are shown in Fig. 7.



a. b.

Fig. 7. Layouts of RCT (a) and RCR (b).

The simulation of the port's extracted schematic demonstrated its operation at data rates up to 1.25Gb/s. The complete port is ready for the integration into higher-level designs as a macro block. Its implementation in the CML basis makes it directly suitable for space-oriented applications which require tolerance to harsh environmental conditions.

6 CONCLUSIONS

A novel technique for transmission of SW-compatible data and clock information through a single differential interconnect line has been developed and implemented as a transceiver test chip and as a SW routing port. Both designs are made within a special CML library of cells and blocks that is developed in a 180*nm* BiCMOS technology.

Utilization of this technology makes the designs suitable for space-oriented applications which require tolerance to harsh environmental conditions.

Fabrication and testing of the transceiver chip has proven the performance of the developed technique.

A novel algorithm and circuitry for the LVDS line integrity control have been developed. This technique allows for the detection of the line's faulty condition and its replacement with the second line defined in the SW standard but not required for the data transmission through the novel three-level interface.

7 REFERENCES

1. David R. Alexander, David G. Mavis, Chatles P. Brothers, and Joseph R. Chavez, “Design Issues for Radiation Tolerant Microcircuits in Space”, 1996 IEEE Nuclear and Space Radiation Effects Conference Short Course, 1996.

2. “Space engineering: SpaceWire - Links, nodes, routers and networks”, ECSS-E-50-12A, 24 January 2003, ECSS Secretariat, ESA-ESTEC Requirements & Standards Division, Noordwijk, The Netherlands, Published by: ESA Publications Division, ESTEC, P.O. Box 299, 2200 AG Noordwijk, The Netherlands, ISSN: 1028-396X.

3. V. Katzman, V. Bratov, “Method and System for Multilevel Serializer/Deserializer”, U.S. Patent No. 7342520, March 11, 2008.

4. “IEEE Standard of Low-Voltage Differential Signaling (LVDS) for Scalable Coherent Interface (SCI)”, *IEEE Std. 1596.3-1996*, March 1996, ISBN 1-55937-746-1.

5. V. Bratov, J. Binkley, V. Katzman, A. Bratov, A. Otero, G. Rakow, “Universal Input Buffer for Programmable Logic Devices”, 9th MAPLD International conference, Washington, DC, September 26-28, 2006.

SPACE-QUALIFIED 1.25GB/S NANO-TECHNOLOGICAL TRANSPONDER FOR SPACE WIRE OPTICAL/ELECTRICAL INTERCONNECTS

Session: SpaceWire Components

Long Paper

Vladimir Katzman, Vladimir Bratov, Glenn P. Rakow

Vladimir Katzman and Vladimir Bratov are with Advanced Science and Novel Technology (ADSANTEC) 27 Via Porto Grande, Rancho Palos Verdes, CA 90230, USA.

Glenn P. Rakow is with NASA Goddard Space Flight Centre, Maryland, USA.

E-mails: vkatzman@adsantec.net, vbratov@adsantec.net, glenn.p.rakow@nasa.gov

ABSTRACT

A space-qualified SW-compatible and Ethernet-compatible Transponder ASIC with 1.25*Gb/s* LVDS serial input/output interfaces and 10-bit wide 125*Mb/s* CMOS parallel input/output interfaces is presented. The chip with power consumption below 150*mW* utilizes a special SCL library of cells and blocks including proprietary extra low-power LVDS buffers. It features a clock multiplication unit on the transmitter side and a clock and data recovery block on the receiver side. Three built-in test modes provide a possibility for a detailed self-testing of the part. The Transponder fabricated in a 90*nm* CMOS technology and packaged in a 64-pin QFN plastic package has demonstrated a reliable operation during laboratory tests.

1 INTRODUCTION

Performance improvement that is required for the success of future space missions dictates the migration from low-speed parallel to high-speed serial data interconnect interfaces. In contrast with ground-based electronics, achievement of high data transmission rates in space-oriented interconnect systems presents a significant challenge due to tight requirements for their stability in harsh operational conditions. Application of the existing techniques for stability improvement, such as the ones described in [1], inevitably degrades the achievable speed-power performance.

Space plug-and-play avionics is an emerging technology that can alleviate serial data interconnect shortcomings in present-day solutions. It is based on the switching fabric active backplane architecture with robust high-speed serial interfaces. Electrical and/or optical transponders operating with Space Wire (SW) [2], optical SW (SW-Fiber), Fire Wire (FW), or Ethernet/Gigabit Ethernet protocols are required to support the associated high-speed data interconnects.

Unfortunately, the achievable performance of the copper-based SW interconnects is limited by the specific structure of the interface that requires two differential channels per unidirectional link to implement the data-strobe (DS) encoding scheme. Unavoidable channel-to-channel skew accompanied by signal degradation during the transmission process complicates the clock recovery process and prevents system operation at high data rates.

All electrical interconnects are also susceptible to EMI, which presents a significant danger to spacecraft's electronics. Taking into consideration the described limitations, one of the most suitable solutions is the replacement of electrical links with fiber-optical lines. This approach requires the development of an electrically DC balanced transponder that interacts with standard optical modules on the high-speed side and standard data processors (e.g. FPGA) on the low-speed side.

This paper presents a design of such transponder with the characteristics detailed in Table 1:

Table 1. System Specifications.

Parameter	Value	Units	Comments
Data encoding type	8B10B		
Parallel interface size	10+1	bits	Data and Clock
Parallel interface type	1.8*V* CMOS		
Low-speed data rate	125	*Mb/s*	
Serial interface type	LVDS		Low-voltage differential signalling [3]
High-speed data rate	1.25	*Gb/s*	
Power supplies	1.8	*V*	±5%
Power consumption	<150	*mW*	

The following sections describe the transponder architecture (Section 2), explain the utilized library-based design approach and discuss the library structure (Sections 3.1), describe chip design (Section 3.2) as well as its fabrication and testing (Section 4).

2 TRANSPONDER ARCHITECTURE

The required functionality can be naturally achieved in a 2-channel serializer/deserializer (SERDES) system shown in Fig. 1. To process the 8B10B data, SERDES includes a transmitter channel (Tx, green blocks in Fig. 1) with 10-to-1 MUX CMU (multiplexer with internal clock multiplication unit), CMOS input parallel interface (CMOS IB), 2-bit FIFO (first-in-first-out), and an LVDS output buffer (OB); as well as a receiver channel (Rx, brown blocks in Fig. 1) with CDR (clock and data recovery unit), 1-to-10 DMUX (demultiplexer), LVDS input buffer (IB), and CMOS output parallel interface (CMOS OB). Two internal phase-locked loops (PLLs) in CMU and CDR synchronize the transponder operation in respect to the external low-speed reference clock "cref". Input parallel data "d00-09" initially aligned to the external input low-speed clock "cli" is resynchronized in FIFO by the internal clock "crd". FIFO reset performed by "res" signal, as well as error indicator "err" are incorporated into the design. To increase the flexibility of parallel interfaces "d00-09" and "q00-09", bit order inversion function controlled by "bitordt" and "bitordr" signals is implemented in both MUX and DMUX.

Three internal loop-back test modes are incorporated into the design. In the first mode activated by "lbt1" signal, the parallel data from DMUX outputs and corresponding low-speed clock "cb1" are redirected to MUX inputs through selector SEL1. In this mode, high-speed serial output signal "qhs" should be compared with high-speed input serial signal "dhs" usually provided from a PRBS (pseudo-random binary sequence) generator. In the second mode activated by "lbt2" signal, the high-speed signal from MUX output is looped back to DMUX input through selector SEL2, and parallel CMOS input and output interfaces are used for the control of transponder operation. The third test mode activated by "lbt3" signal is similar to the second test mode but provides a by-pass of CDR. In this mode, the output data

from MUX and corresponding clock “cb3” are delivered to DMUX inputs through selector SEL3.

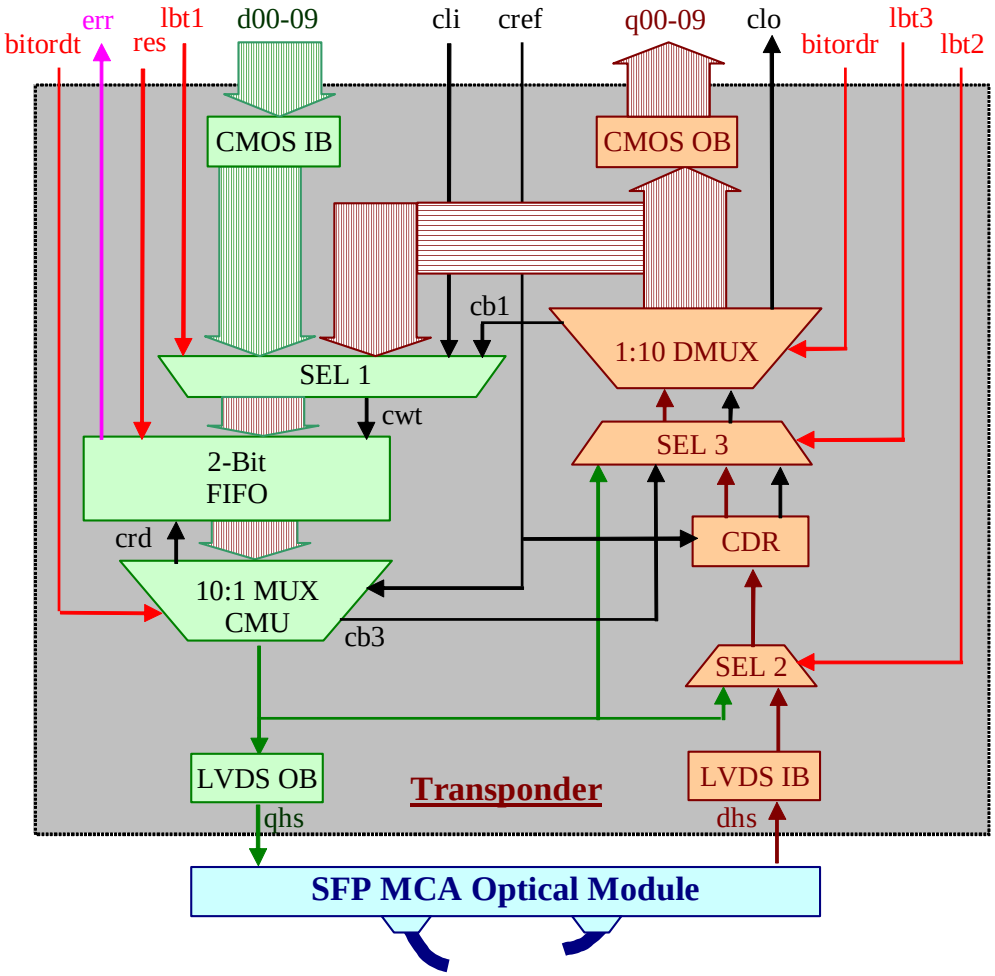


Fig. 1. Transponder Top-Level Block Diagram.

3 Library-Based Design Approach

The complete SERDES is designed within a library of basic cells and functional blocks that has been developed in a 90*nm* CMOS technology. This approach allows for application of previously verified cells and block in new products, which significantly reduces their cost and design time. All cells and blocks in the developed library utilize custom-built transistor structures that provide an improved tolerance to space environmental conditions.

3.1 Library of Basic Cells

CMOS logic is a natural implementation of Si-based CMOS technologies [3]. This single-ended architecture based on FETs with gate lengths of 90*nm* and below is now utilized in most of the modern IC products. At the same time, the short-channel devices suffer from increased sub-threshold currents and low breakdown voltages. They also cannot overcome the natural drawbacks of the single-ended circuitry that includes generation of a switching noise that is increasing with the operational speed, and distortion of the signal’s duty cycle. In addition, uutilization of space-oriented protection techniques usually limits the minimum transistor width and thus negatively affects the operational speed of the corresponding

circuits. It should be noted that the absence of DC power consumption does not help the overall performance of the CMOS circuits due to significant increase of dynamic power consumption at higher speeds. In this particular design, analog blocks represented by two PLLs are very sensitive to any source of noise and require a quieter on-chip environment.

The alternative for CMOS is the differential source-coupled logic (SCL) architecture [5-6]. This logic is based on differential current switches with one or more pairs of switching n-FETs which redistribute a stabilized current generated by a special current source. SCL architecture overcomes the problems of the switching noise and duty cycle distortion at the price of certain DC power consumption. SCL circuits are less sensitive to sub-threshold currents and can potentially achieve higher operational frequency. Unfortunately, the limited transconductance of FETs minimizes the architectural advantages due to the requirements for a higher voltage logic swing (not less than $350mV$) to fully switch the tail current from one branch of the switch to the other. The driving capability of SCL logic gates is also relatively low, thus resulting in increased power consumption. This architecture is suitable for the operational speeds in the range of $10Gb/s$, but its performance degrades rapidly at higher frequencies.

The comparison of two architectures was performed on two sets of frequency dividers-by-10 designed and fabricated in the same $90nm$ technology. The test results summarized in Table 2 show that the power consumption of the CMOS version is close to that of the SCL one at frequencies close to $700MHz$. It can be estimated that the SCL version is more power efficient at speeds above $1GHz$. In addition, the SCL architecture is more suitable for the implementation of complex logic functions as can be seen from the number of gates required for the divider design.

Table 2. Divider Comparison

Parameter	***CMOS Version***	***SCL Version***
Operational frequency	$700MHz$	$720MHz$
Power consumption	$1.2[V]\cdot 0.6[mA]=0.72[mW]$	$1.6[V]\cdot 0.5[mA]=0.8[mW]$
Gate count	61	21

Based on the above considerations, the SCL architecture has been selected for the transponder design. The developed library includes 11 families of cells with different current levels optimized for frequencies from DC to more that $2GHz$: 2-input AND gates, 1-input Buffers with and without input-level compensation diodes, 2-input dual Source Followers – level shifters, D-Latches with and without set/reset functions, 2-to-1 Multiplexers, RS Flip-Flops, Generators of the top logic "0" voltage level $V^{0t}=V_{CC}-\Delta V_L$, 2-input XOR gates, and 2-input SCL-to-CMOS converters.

All cells operate from a $1.8V$ power supply with tail currents generated by two-transistor cascode current sources controlled by temperature-stabilized band-gap reference sources. The achieved voltage accuracy is within ±1.5% over the temperature range of -55°C to 125°C, ±10% variation of supply voltage, and ±3σ variation of technological parameters. An example of the designed cell is shown in Fig. 2. This stabilization allows for the minimization of the internal voltage swing to $350mV$, which improves the speed performance of the circuits. As a result, a cell with a current level of $280\mu A$ can reliably process signals with a frequency up to $1.3GHz$.

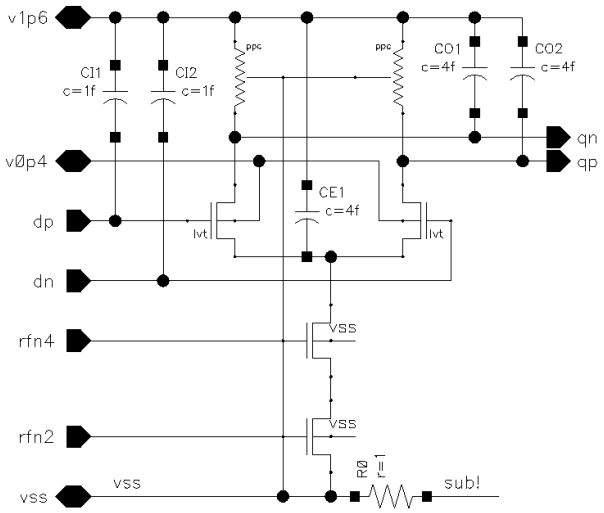


Fig. 2. SCL Buffer.

The library also includes extra low-power 1.3*Gb/s* LVDS output buffers and universal input buffers with increased tolerance to common-mode voltage variation, which are described in [7-8]. The output buffer operates from a 1.6*V* supply voltage generated by an internal voltage source with feedback stabilization.

3.2 Upper-Level Blocks

The most critical parts of the transponder are CDR and CMU. CDR uses the architecture with a phase – frequency detector (PFD), Alexander phase detector (PD), and 2-input charge pump as shown in Fig. 3.

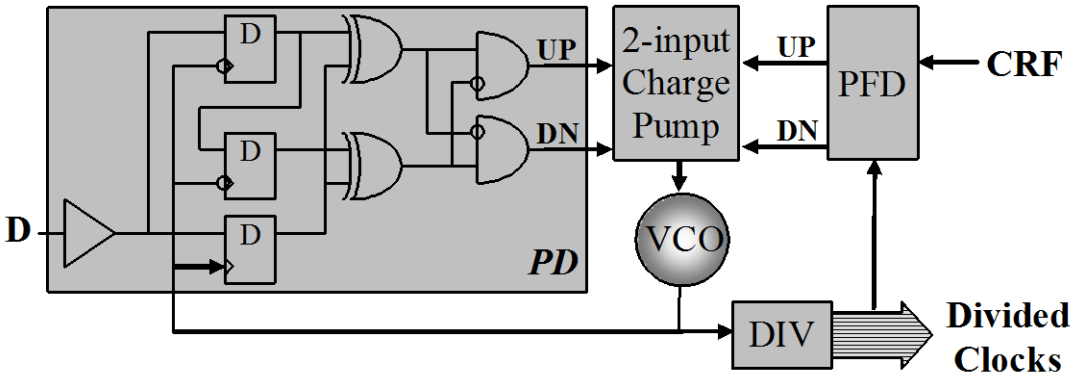


Fig. 3. CDR Architecture.

The 1.25*GHz* voltage-controlled oscillator (VCO) is designed as a ring oscillator with balanced loadings to ensure matching operational modes for all stages of the ring. The tuning range from 950*MHz* to 1.5*GHz* has been achieved with 5-stages of a proprietary voltage-controlled delay cells. The sell is designed as SCL D-type latch with a linearized clock differential pair that operates as the delay control current switch. The schematic of the delay cell is shown in Fig. 4.

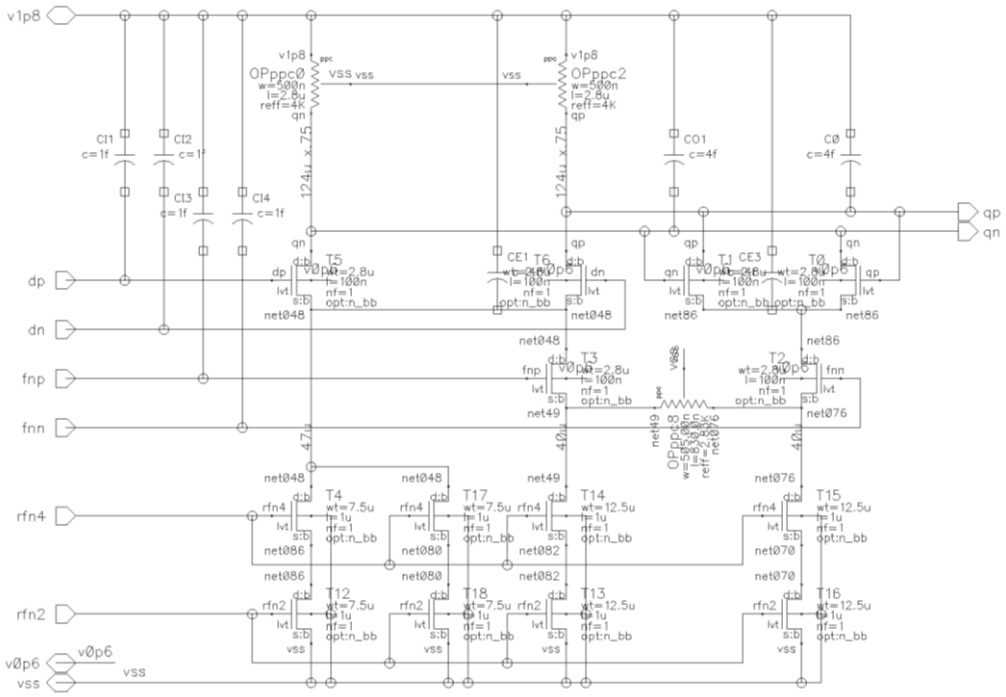


Fig. 4. Voltage-Controlled Delay Cell.

The CMU utilizes similar VCO and PFD blocks but a different 1-input charge pump. All other blocks are designed from standard library cells.

MUX and DMUX utilize a combinational tree-register architecture due to the ratio not equal to a power of 2.

FIFO includes 10 identical blocks for processing 10 bits of data and an error generation block. Each FIFO block writes in a bit of data using the input clock "cwt" and outputs the same bit after its realignment to the output "crd" clock. The initial reset of the block sets the read and write events as far as possible on the time scale and then allows for $\pm(T\text{-}t_{su})$ phase deviation between two clocks. Here T is the clock period and t_{su} is a setup time of internal flip-flops. If the accumulated phase difference exceeds the specified value, the error generator provides a special error signal. This block operates as a phase detector and monitors the phase difference between two clocks. The phase difference equal or less than t_{su} sets the block's output to logic "1" state that indicates the error condition. Assuming slow phase changing rate (temperature or similar conditions), the block is designed to deliver the error signal before any actual errors in the data sampling may occur.

4 CHIP FABRICATION AND TESTING

The complete transponder has been fabricated in a 90*nm* technology. The chip with the dimensions of 2.4x2.4mm^2 has 68 bonding pads, has been packaged into a 68-pin QFN package, and consumes about 80*mA* of current from a single 1.8*V* power supply.

The test board shown in Fig. 5 incorporates the transponder chip, an SFP MSA optical module, SMA connectors for low-speed and high-speed data and clock signals, and control switches. The device was tested in electrical and optical modes and demonstrated the operation in accordance with computer simulations.

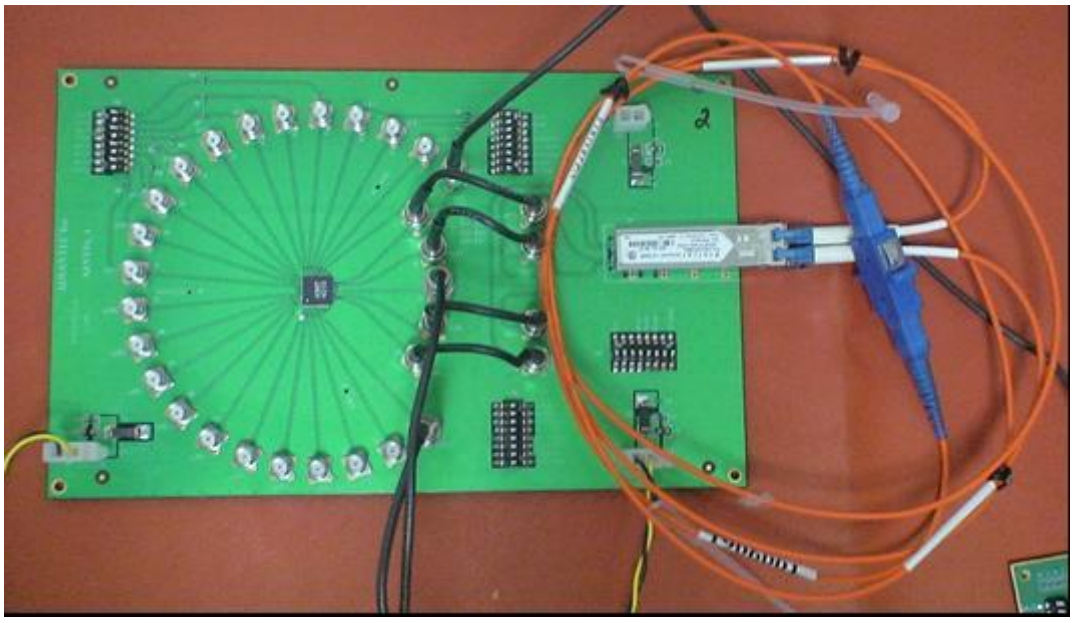


Fig. 5. Transponder Evaluation Board Configured For Optical Testing.

The sample results of the electrical test shown in Fig. 6 demonstrate the output low-speed clock and serial high-speed data with the rate of 1.25*Gb/s*.

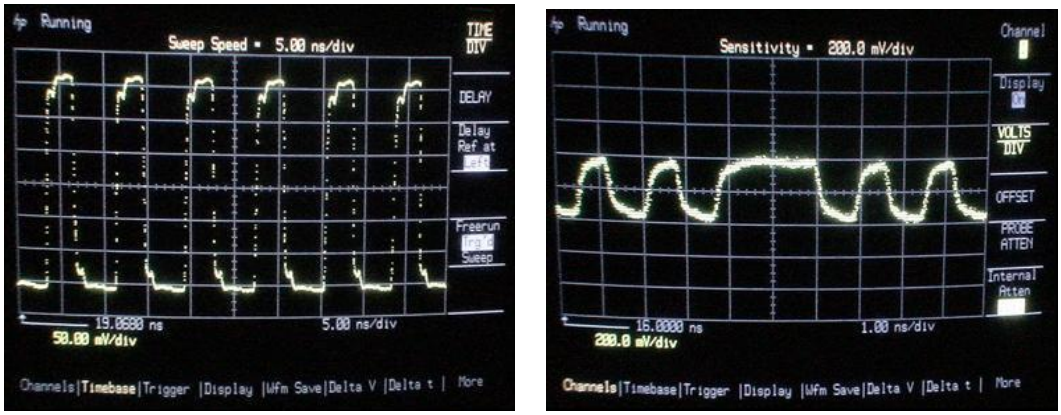


Fig. 6. Output Low-Speed Clock (a) and 1.25Gb/s Output Data (b).

In the optical mode, the output LVDS signal of the Transmitter is converted into light by the optical module and sent through the fiber back to the optical input of the same module. The converted electrical signal is processed by the Receiver and the outputs of the receiver are compared to the Transmitter input signals. This external loop-back test has proved the complete functionality of the designed transponder.

The chip was also tested in the space-imitation environment and demonstrated less that 1% deviation of its parameters from their normal values.

5 CONCLUSIONS

A space-qualified SW-compatible transponder chip designed and fabricated in a 90*nm* technology has demonstrated a full functionality in normal and space-imitation environments including the data rate of 1.25*Gb/s* at less than 150*mW* power consumption. The design is based on a special SCL library of cells and functional blocks utilizing *n*-FETs as active components, which is ready for application to other designs of similar mixed-signal products.

6 REFERENCES

1. David R. Alexander, David G. Mavis, Chatles P. Brothers, and Joseph R. Chavez, “Design Issues for Radiation Tolerant Microcircuits in Space”, 1996 IEEE Nuclear and Space Radiation Effects Conference Short Course, 1996.

2. “Space engineering: SpaceWire - Links, nodes, routers and networks”, ECSS-E-50-12A, 24 January 2003, ECSS Secretariat, ESA-ESTEC Requirements & Standards Division, Noordwijk, The Netherlands, Published by: ESA Publications Division, ESTEC, P.O. Box 299, 2200 AG Noordwijk, The Netherlands, ISSN: 1028-396X.

3. “IEEE Standard of Low-Voltage Differential Signaling (LVDS) for Scalable Coherent Interface (SCI)”, *IEEE Std. 1596.3-1996*, March 1996, ISBN 1-55937-746-1.

4. R. Baker, H. Li, D. Boyce “CMOS Circuit Design, Layout, and Simulation”, IEEE Press, ISBN 0-7803-3416-7, 1998, pp. 199-288.

5. E.Bushehri, V. Bratov, V.Staroselski, “Loading element for a logic gate”, U.S. Patent No. 5,920,205, July 6, 1999.

6. E.Bushehry, V.Bratov, V.Staroselsky, T.Schlichter, S.Milenkovic and V.Timoshenkov, “Ultra-Low Power Source Coupled FET Logic gate configuration in GaAs MESFET Technology, Electronics Letters, vol.36, No. 1, January 2000, p.36-38.

7. V. Bratov, J. Binkley, V. Katzman, and J. Choma, “Architecture and Implementation of a Low-Power LVDS Output Buffer for High-Speed Applications”, IEEE Trans. on Circuits and Systems I, v. 53, No. 10, Oct. 2006, pp. 2101-2108.

8. V. Bratov, J. Binkley, V. Katzman, A. Bratov, A. Otero, G. Rakow, “Universal Input Buffer for Programmable Logic Devices”, 9th MAPLD International conference, Washington, DC, September 26-28, 2006.

EXPANDING PORT COUNT USING A 4-PORT SPACEWIRE ROUTER

Session: SpaceWire Components

Short Paper

Jennifer Larsen

Aeroflex Colorado Springs

4350 Centennial Blvd. Colorado Springs, CO 80907

E-mail: *Jennifer.Larsen@aeroflex.com*

ABSTRACT

The UT200SpW4RTR[2] is a four port SpaceWire router that is capable of operating at data rates from 10 to 200 Mbps, supporting path, logical, and group adaptive routing and offers an effective and simple solution to many networking requirements. This router has a total of 5 ports, 4 are SpaceWire compatible ports and the 5th is a parallel HOST port.

The HOST port allows access to the routers configuration and status registers as well as access to any of the 4 SpaceWire ports. Data may be passed from the HOST port to the SpaceWire ports and vice versa. HOST ports of multiple UT200SpW4RTR devices may be interfaced together using an FPGA which will route data between multiple routers. The interfacing FPGA will need to contain logic that will support reads and writes from the UT200SpW4RTR HOST ports as well as a lookup table block. The look up table block will contain the routing information such that data can be passed to and from the HOST ports of multiple routers.

1 UT200SPW4RTR BASIC FUNCTIONALITY

The Aeroflex 4-port router implements a non-blocking crosspoint switch and a "Round Robin" arbitration scheme allowing all 5 receive ports access to all 5 transmit ports. Path and logical addressing are supported per ECSS-E-ST-50-12C [1], and lookup table storage is replicated five times giving each receive port a dedicated block of memory for logical addressing. Configuration of lookup tables, as well as access to internal registers may occur through any of the 5 ports using a simple configuration protocol. A group adaptive function is also provided for 2 ports when implementing logical addressing.
Each of the four SpaceWire ports is capable of running at an independent speed. This allows for systems to be configured with nodes/instruments running at different speeds.

The HOST port of the 4-port router is composed of both a receive and transmit FIFO. The transmit FIFO (inputs to router) are write capable by the external hardware. Full and Almost Full flags are provided to help the user prevent overwriting the FIFO and should be monitored by external hardware or the interfacing FPGA. Data will be

written into the FIFO on the rising edge of the clock when /TX_PUSH is “Low”. The receive FIFO (outputs from router) receives data from one of the SpaceWire ports and is then read from the receive FIFO on the rising edge of the system clock when /RX_POP is “High”. FIFO status flags Almost Empty and Empty flags are provided for proper data management.

The HOST port transmit interface is connected to a read logic block that controls SpaceWire data flow and determines the addressing scheme being used for the packet received, where as the HOST port receive interface is connected to a write logic block to the receive FIFO interface. Each of the SpaceWire ports on the UT200SpW4RTR contain a read logic block. The basic concept of the read and a write logic blocks should be replicated in the FPGA.

Read logic blocks are connected to each of the SpW ports as well as a the HOST port. The SpaceWire ports read logic block have internal FIFO monitor flags that can not be accessed externally. The read logic block monitors the empty flag on the receive FIFO and reads a byte of data whenever the FIFO is not empty. This block also checks the first byte of data after an EOP to determine the port address or whether a configuration transaction will be initiated. For Path or Logical addressing, the Read Logic Block uses the first byte of data after an EOP/EEP.

The write logic blocks control the data to the transmit FIFOs and the HOST receive port and the SpW transmit ports. A "Round Robin" arbiter manages access and makes sure only one Read Logic Block accesses the Write Logic Block. If more than one receive ports is waiting to send data out of the same output port, the arbiter gives each receive port equal opportunity for access.

The arbiter starts counting whenever a request for that port is received from any of the five receive ports. The count is from Port 1, Port 2, etc, until the count reaches Port 5, looks for configuration commands, and then starts over. The configuration block will be accessing the Write Logic Block when read configuration packets are requested. The HOST port of the UT200SpW4RTR will allow the system designer to interface multiple 4-port routers together with out compromising the count of the SpaceWire capable ports.

2 SYSTEM ARCHITECTURE

Figure 2 shows a notional diagram using three UT200SpW4RTR routers interfaced to an FPGA, a microprocessor could be controlling the system. This example generates a 12-port router using three 4-port routers. These concepts can be applied to generate a router with a larger port count.

The interfacing FPGA should contain a look up table space that is responsible for routing data between the HOST ports of the 4-port routers, as well as, read and write logic blocks as described insection1.0. The look up table space should be configured such that a given logical address will be routed to the HOST port of the destination 4-port router. The look up space can be sized based on the number of 4-port routers being interfaced to the FPGA.

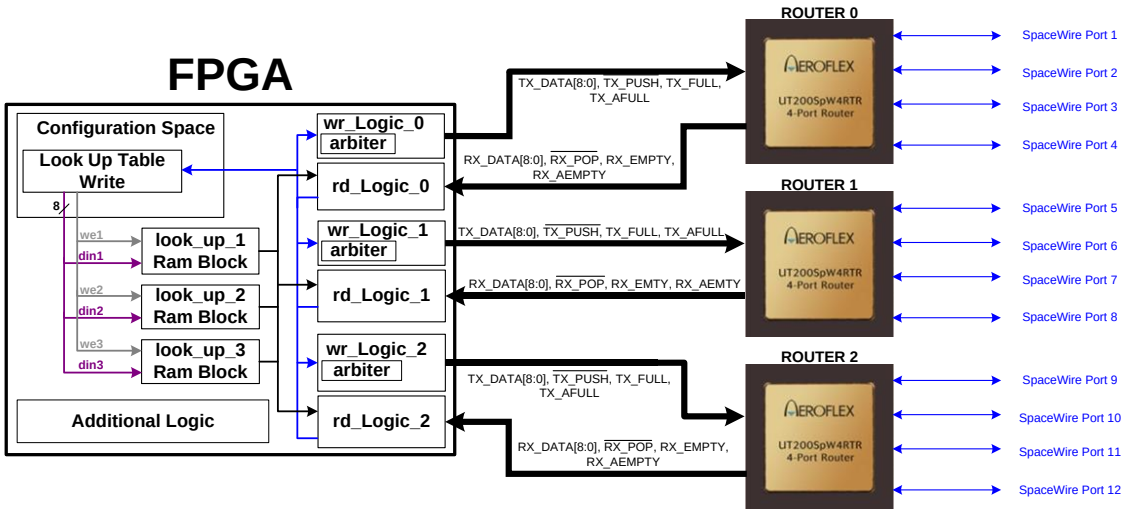


Figure 2. Notional Diagram of FPGA requirements

The HOST port interfaces of the UT200SpW4RTR devices, including TX_DATA[8:0], /TX_PUSH, TX_FULL, TX_AFULL, RX_DATA[8:0], /RX_POP, RX_EMPTY, RX_AEMPTY, should be connected to the FPGA for data transfer and status monitoring. The read and write logic blocks will control the flow of data from one HOST port to the other. Figure 3 shows a block diagram of the main logic block requirements needed to handle arbitration, routing, and data transfer in the FPGA.

Table 1 shows an example of the contents required for the FPGA look up table space. Each of the 12 SpW ports and 3 HOST ports requires a unique look up table location, to ensure proper data routing. Using Table 1, assume SpaceWire port 6 (port 2 on Router 1) has data that needs to be routed to SpaceWire port 11 (port 3 on Router 2). A packet with header 0x30 is sent to Port 6. Router 1 decodes lookup table address 0x30 and sees that data should be sent to the HOST port or local port 5 of Router 1. The FPGA rd_Logic_1 read logic block decodes packet header 0x30 and sees that data should be sent to Router 2 wr_logic_2. Router 2 HOST port decodes lookup table address 0x30 and sees that data should be sent to Port 11 (port 3 on Router 2). The data packet will be routed out on Port 11 of the expanded router.

Table 1. Example Lookup Table Space.

Address	Router 0	Router 1	Router 2	FPGA	SpW Port
0x20	1	HOST	HOST	1	1
0x21	2	HOST	HOST	1	2
0x22	3	HOST	HOST	1	3
0x23	4	HOST	HOST	1	4
0x24	HOST	1	HOST	2	5
0x25	HOST	2	HOST	2	6
0x26	HOST	3	HOST	2	7
0x27	HOST	4	HOST	2	8
0x28	HOST	HOST	1	3	9
0x29	HOST	HOST	2	3	10
0x30	HOST	HOST	3	3	11
0x31	HOST	HOST	4	3	12

3 SYSTEM PERFORMANCE CONSIDERATIONS

The routing of SpW data to and from each of the router devices should follow a flow similar to that shown in Figure 3. The flow in Figure 3 may be modified to optimize data through put and overall system efficiency.

The flow of data to the interfacing FPGA can start with the FPGA in idle state, where the /RX_POP FIFO flag is being monitored for incoming data. Once a /RX_POP FIFO flag is asserted active low, the FPGA will decode which 4-port router has the active /RX_POP flag. After the active router is identified the FPGA starts registering the data present on the RX_DATA[8:0] lines until an End of Packet (EOP) is received.

When an EOP is received the first byte of data needs to be examined. The first byte of data should contain the logical look up information as described in Table 1. Byte 1 will then be compared to the FPGA Logical Look up table address such that the data will be routed to the correct destination 4-port router.

Assuming the first byte of data contains a valid router address the FPGA needs to monitor the corresponding 4-port routers TX_FULL to ensure that there is available space in the routers HOST transmit FIFO. The FPGA then asserts the corresponding 4-port routers /TX_PUSH FIFO flag, this starts the transfer of data from the FPGAs register to the destination 4-port router. Once an EOP has been received by the destination 4-port router the FPGA returns to the idle state and wait for the next data transaction to occur.

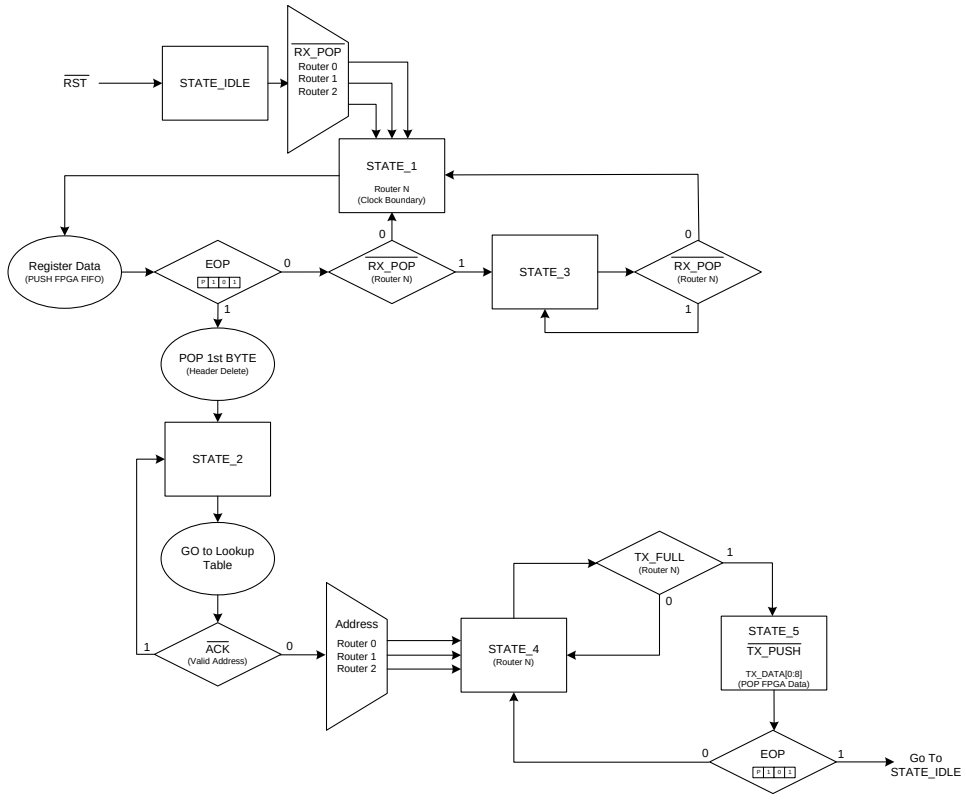


Figure 3. FPGA Data Handling Flow Diagram

4 CONCLUSION

Interfacing multiple UT200SpW4RTRs together using the HOST port offers a simple solution to increase port count. The HOST ports should be interfaced together using a FPGA that will act as an arbiter, equipped with look up tables, between the multiple UT200SpW4RTR devices.

The interfacing FPGA needs to contain logic supporting reads and writes to and from the HOST ports read and write FIFOs. There will be a look up table block which contain the routing information such that data can be passed to and from the HOST ports of multiple routers.

5 REFERENCES

[1] ESA Publications Division, "SpaceWire Standard Document ECSS-E-ST-50-12C", The Netherlands, July 30, 2008.

[2] Aeroflex, "UT200SpW4RTR 4-port SpaceWire Router Datasheet", Colorado Springs, Colorado, February 2010.

Components 2

ASTRIUM'S SPACEWIRE BASED LEON PROCESSORS

Session: SpaceWire Components

Short Paper

Paul Rastetter, Tim Helfers, Mark Hartrampf

Astrium GmbH, 81663 Munich, Germany

Jean-Luc Poupat

Astrium SAS, 78997 Elancourt, France
E-mail: paul.rastetter@astrium.eads.net, tim.helfers@astrium.eads.net, mark.hartrampf@astrium.eads.net, jean-luc.poupat@astrium.eads.net

ABSTRACT

There are two SpaceWire based processors in Astrium. One is named SCOC3 and one is named MDPA. SCOC3 uses LEON3-FT and MDPA uses LEON2-FT, both processors are based on SPARC V8 standard.
The main application of SCOC3 is platform while MDPA's main application is payload. The paper describes the different interface modules used to achieve the functionality and also the commercialisation aspects of the two processors.
MDPA (Multi-DSP/micro-Processor Architecture) is a highly integrated System-on-Chip system, which is an advancement on the architecture developed and used on the Inmarsat4 DSP payload. The MDPA is a concept based on a matrix of data processing nodes interconnected using SpaceWire (SpW), with external SpW interfaces to enable connection to other telecommunication, earth observation or science payload subsystem. An MDPA node is in effect a system-on chip which incorporates a highly integrated DVB-S modem co-processor combined with a powerful LEON2-FT microprocessor function and relevant interfaces all integrated on the same device. This architecture acts as the controlling unit for the Data Path Subsystem (DPS) within the frame of the next generation of digital telecommunication payloads. Additionally the MDPA concept is laid out for high end control applications or medium rate data processing for earth observation or science payloads.
The 8 SpaceWire interfaces can be used to interconnect several MDPA nodes to a multi-processor configuration. This increases the overall processing performance and enhances the processing redundancy since a faulty node can be replaced by another one. The routing capabilities support the communication of the nodes with low processor interaction. In addition the high number of SpaceWire interfaces allows connection of several remote controlled devices for command and monitoring of subsystems.
The modem function is implemented as hardwired block on-chip. Other devices such as GNSS receivers or reconfigurable co-processors can be used externally and controlled via SpaceWire.

SCOC3 is based on 7 independent AMBA controllers programmed individually by the SPARC processor. They work by using DMA mechanisms to access a specific memory and to free the processor. They are allocated to CCSDS communications, I/O User communication, Reconfiguration of satellite's system and tests links.
All these SpaceWire links are compliant to the last ECSS standard and are RMAP compatible.

MDPA Technical Overview

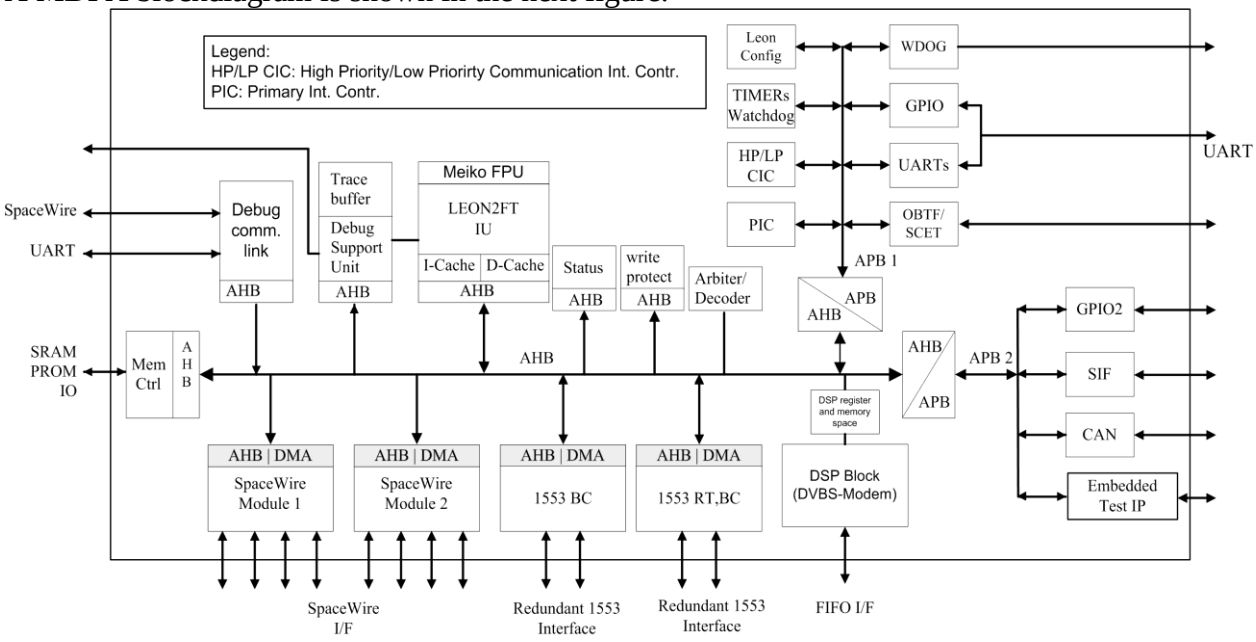
MDPA is a highly integrated System-on-Chip system. This architecture offers the benefits of efficiency and the savings provided by a hardwired DSP and the flexibility offered by a programmable microprocessor system, joint together using state-of-the-art interconnection IP.
The main tasks of MDPA are

- (de-)modulation and (de-)coding capabilities for regeneration of telecommand and generation of telemetry channels
- Supervise, configure and monitor the Data Path Subsystem (DPS)
- Interface with spacecraft central computer

The functionality is provided by means of the following functions, integrated into a single chip.

- The fault tolerant microprocessor core LEON2-FT
- A digital signal processing module
- Various interfaces (SpaceWire with routing capabilities, MilBus, CAN bus)

A MDPA blockdiagram is shown in the next figure.



MDPA Processor core

The processor core is based around the LEON2-FT using one AMBA AHB bus and two APB buses. The APB buses are connected via bridges to the AHB. The following main communication interfaces are provided

- 8 SpaceWire links
- 2 redundant MILBUS
- 1 CAN bus

One set of SpaceWire interfaces are used to interconnect several MDPA nodes to a multi-processor configuration. This increases the overall processing performance and enhances the Fault Detection, Isolation and Recovery (FDIR) capabilities since a faulty node can be replaced by another one. The other SpaceWire interfaces are routed to the Data Path Subsystem carrying the switches and beamformer configuration data.
The MDPA contains a CAN bus interface for low data rate data transfers to nodes and peripherals. The Service Interface (test interface) based on SpaceWire is used for advanced software debugging support as memory load commands etc. with 100 Mbps data rate.

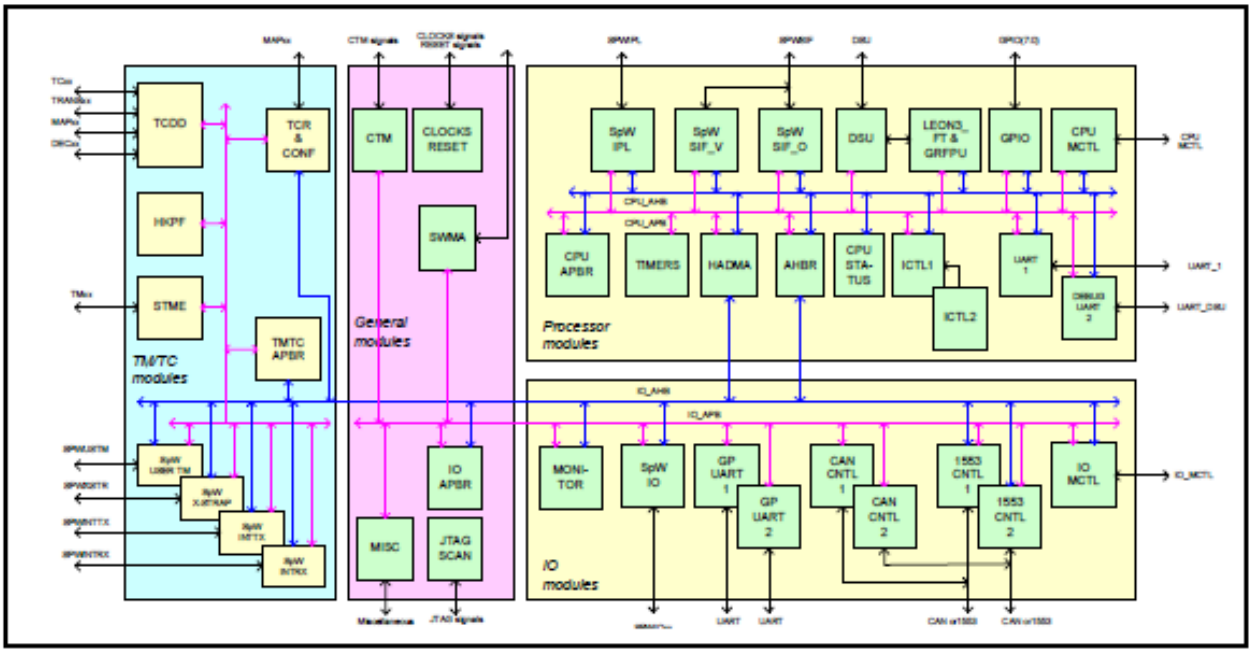
MDPA DSP module

The DSP module is equipped on each MDPA node and demodulates and decodes the incoming time-division multiplex (TDM) telecommands coming from the Network Control Center. Additionally the data dedicated for downlink telemetry are coded and modulated.

SCOC3 Technical Overview

The Satellites are controlled via a platform computer that permits the control of the satellite (attitude, orbit, modes, temperatures …) with respect to its payload mission (communication, earth observation, scientific mission). The platform computer is connected to the satellite and the ground control via digital links and executes onboard software. On a hardware point of view, it gathers a lot of digital functions, usually dispatched on numerous devices, in a single device.

At architectural level there are two memory interfaces, one dedicated for the LEON3-FT and another one for the peripheral IOs (SpaceWire, 1553 mil-bus, TM/TC) and an AMBA architecture/DMA for the performance.



A SCOC3 blockdiagram is shown in the next figure.

SCOC3 Processor core

SCOC3 is based on LEON3-FT. There are two AMBA AHB buses, one dedicated to the processor and one to interfaces functions. Both are linked by a bridge. They are used to transfer data. There is an APB bus for configuration of functions. APB and AHB are linked by a bridge.

The following main communication interfaces are provided

- 7 SpaceWire links
- 2 redundant MILBUS
- 2 CAN bus

The 7 SpaceWire links are allocated to different functions. Two are allocated to TM/TC, two for cross-strapping and the others are user's free.

The implemented CCSDS TM/TC functions are dedicated to satellite/ground communications through the satellite transponder. These functions work either with the processor or in stand alone. There is a specific CCSDS TM/TC area dedicated to communication between the satellite and the ground through the satellite transponder.

Technology and Packaging

MDPA and SCOC3 are realised as ASICs in Atmel's space qualified 180 nm technology, called ATC18RHA. MDPA is delivered in a CQFP 352 pin package and SCOC3 in a 472 pin Column Grid Array package (CGA).

HIGH PERFORMANCE PPC BASED DPU WITH SPACEWIRE RMAP PORT

Session: SpaceWire Components

Short Paper

Pekka Seppä, Petri Portin

Patria Aviation Oy, Systems/Space, Naulakatu 3, FI-33100, Tampere, Finland

Omar Emam

ENS - Future Mission Systems, EADS Astrium, Gunnels Wood Road, Stevenage - SG1 2NJ, England - UK

Wahida Gasti

European Space Agency, Postbus 299, NL-2200 AG Noordwijk, The Netherlands

Chris McClements, Steve Parkes

University of Dundee, School of Computing, Dundee, Scotland, UK

E-mail: pekka.seppa@patria.fi, petri.portin@patria.fi, omar.emam@astrium.eads.net, wahida.gasti@esa.int, chris@star-dundee.com, sparkes@computing.dundee.ac.uk

ABSTRACT

This paper presents a high performance Data Processing Unit (DPU) with a SpaceWire RMAP Interface. Over 1000 MIPS / 800 MFLOPS performance at 800 MHz CPU clock is achieved with a high performance PowerPC CPU. The DPU has three SpaceWire links operating at up to 100 Mbits/s, two of which with RMAP. These are used to connect the DPU to other SpaceWire nodes. The DPU's 100 Mbits/s RMAP ports can be RMAP Initiator, an RMAP Target or both. The DPU has been developed for use on the MARC demonstrator.

The SpaceWire RMAP port of the DPU is implemented by using ESA's SpaceWire RMAP IP Core. This allows that the RMAP port of the DPU supports both the Initiator RMAP Interface and Target RMAP Interface. The VHDL RMAP IP Core is integrated in the DPU system by VHDL user logic which allows CPU / user software to access all the RMAP IP functions and generates interrupts for the CPU for fluent software execution.

The RMAP IP Core along with the CPU bus interface, memory interfaces with DMA, UART and Interrupt controller are implemented in Actel's Axcelerator AX2000 and ProASIC3E A3PE3000 FPGAs.

1 INTRODUCTION

1.1 CPU

Central Prosessing Unit (CPU) of the DPU is MPC7448, a member of PowerPC 7450 RISC Microprocessor family. The processor has a superscalar architecture that can

dispatch and complete three instructions simultaneously, and thus its performance is typically higher than one MIPS / MHz. The CPU contains several execution units including an IEEE-754 compliant double-precision Floating Point Unit (FPU), 32 kbyte Level 1 (L1) instruction and data caches, and a 1 Mbyte Level 2 (L2) cache memory [1]. The MPC7450 implements the 32-bit portion of the PowerPC architecture, which provides 32-bit effective addresses, integer data types of 8, 16, and 32 bits, and floating-point data types of 32 and 64 bits. The CPU interfaces to memory and peripherals by a 64-bit wide bus using 60x protocol [2].

The CPU is set to operate at 800 MHz internal clock frequency in the DPU, although CPU type selected could be clocked up to 1420 MHz. This decision has been made in order to reduce power consumption and avoid thermal cooling by a fan.

1.2 RMAP ON SPACEWIRE

SpaceWire is a communications network for use onboard spacecraft. It is designed to connect sensors, memories, data processing units and a downlink telemetry subsystem providing an integrated onboard data-handling network. SpaceWire links are serial, high-speed (2 Mbits/s to 200 Mbits/s or higher), bi-directional, full-duplex, point-to-point data links that connect together SpaceWire equipment. Application information is sent along a SpaceWire link in discrete packets. Control and time information can also be sent along SpaceWire links. SpaceWire is defined in the European Cooperation for Space Standardization (ECSS) ECSS-E50-12C standard [3].

There is a number of communication protocols that can be used in conjunction with the SpaceWire. The Remote Memory Access protocol (RMAP) is one of these protocols and is specified in ECSS-E-ST-50-52C standard [4]. The aim of RMAP is to support reading from and writing to memory in a remote SpaceWire node. RMAP can be used to configure a SpaceWire network, control SpaceWire nodes, and to transfer data to and from SpaceWire nodes. RMAP can also be used to download and debug software on a remote processor.

DPU's RMAP SpaceWire links are implemented using the ESA SpaceWire - RMAP IP core developed by University of Dundee for ESA. There are two main function types of the RMAP IP core. The first type is referred to as the Initiator RMAP interface, which sends out RMAP commands and receives any replies. The second type is the Target RMAP Interface, which receives RMAP commands, executes them and sends out any required replies [5].

1.3 DPU

MARC demonstrator DPU is a general purpose 800 MHz high performance space flight (payload) computer providing 1 Dhrystone MIPS (DMIPS) / MHz performance. The DPU high-speed interface to external world is three SpaceWire links, two of which support RMAP protocol.

The breadboard model of the DPU computer will function as a software development board and a demonstrator in MARC SpaceWire - RMAP network. Breadboard models were manufactured using mainly commercial quality grade components. Industry- and military-grade components may be used as well in places where commercial

equivalent is not suitable to represent the required function. In general all the core DPU components in the design has space flight equivalent counterparts.

2 DPU ARCHITECTURE

The system architecture of the MARC demonstrator DPU is depicted in **Figure 1**. The DPU system consists of the CPU, System Control Unit (SCU), SpaceWire Control Unit (SpWCU) and memory. DPU's main external data interfaces are the three SpaceWire links capable to 100 Mbits/s data rate full-duplex. The main software debug interface is an EIA-232 Universal Asynchronous Receiver / Transmitter (UART), and the PowerPC COP interface is for low-level software debugger connection from a remote computer. DPU boot-loader and self-test software are controlled via UART.

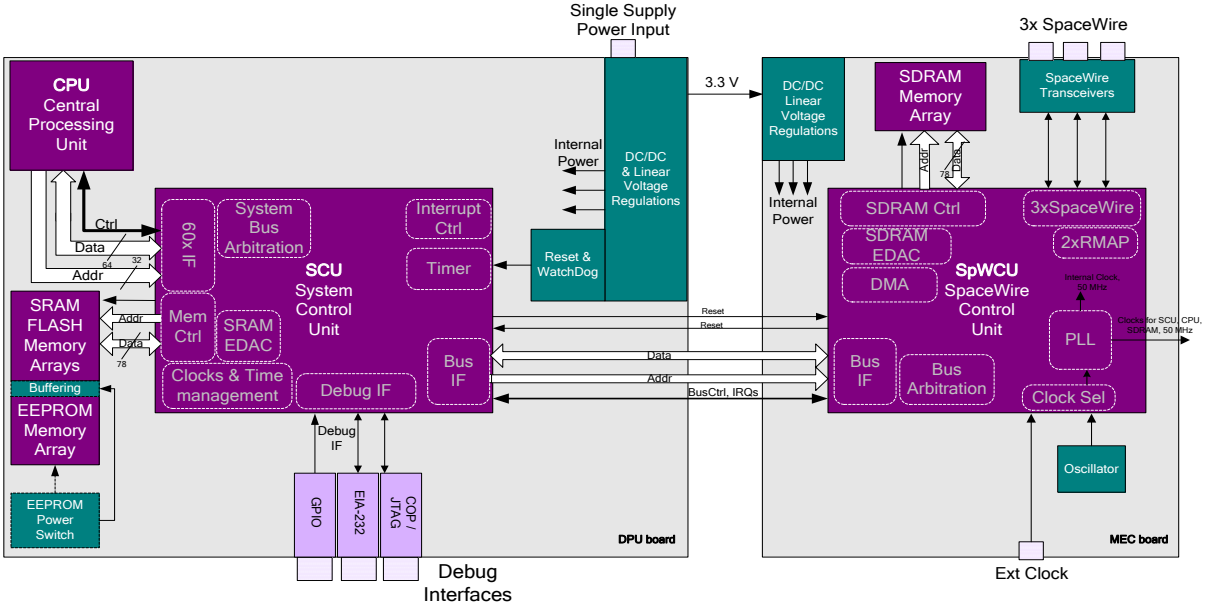


Figure 1 Block Diagram of the DPU

The SCU interfaces the processor with 64-bit PowerPC 60x bus to DPU memory and functions. The SCU controls SRAM, FLASH and EEPROM memory interfaces, and timer, interrupt controller, GPIO, UART and SRAM EDAC are embedded in the SCU. The SCU interfaces also the SpWCU and the CPU by a custom made 32-bit bus.

The SpWCU controls the SDRAM memory interface, and interfaces the Unit to the CPU via SCU. Embedded Direct Memory Access (DMA) unit controls accesses from the CPU and SpaceWire / RMAP functions to the SDRAM memory. The SpWCU and SDRAM memory devices are placed on a separate mezzanine board from the DPU main board.

The DPU memory consists of the 256 kB FLASH for boot program storage, 4 MB EEPROM which can be unpowered when not used, 8 MB EDAC protected SRAM and 512 MB EDAC protected SDRAM. All the memory is accessible by the CPU and can act as storage either for code or data. Data transfers between the DPU and external world through SpaceWire links is routed via SDRAM.

2.1 DMA CHANNEL FOR SPACEWIRE

Purpose of the DMA is to transfer data packets directly between SpaceWire interfaces and SDRAM memory in the dedicated bus independently of the CPU. Meanwhile the CPU can perform other tasks using its local cache or access any other memory space than SDRAM as is illustrated in **Figure 2**. Each SpaceWire receiver and transmitter has it's own DMA channel to SDRAM memory.

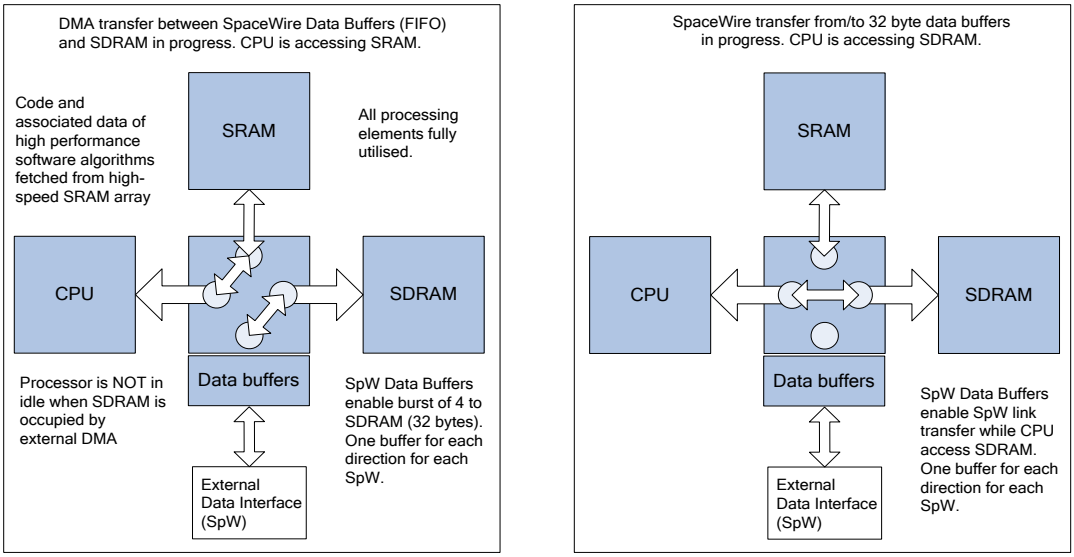


Figure 2 DMA to SDRAM memory

In order to exploit fast data rate SDRAM burst accesses, received data is collected in 32 bytes (4 x 64-bit) buffer before written in SDRAM, and data to be transmitted is read in 32 bytes buffer from SDRAM. SpaceWire data interface buffers are depicted in **Figure 3**. Theoretical data rate is 229 Mbytes/s for SpaceWire access to SDRAM.

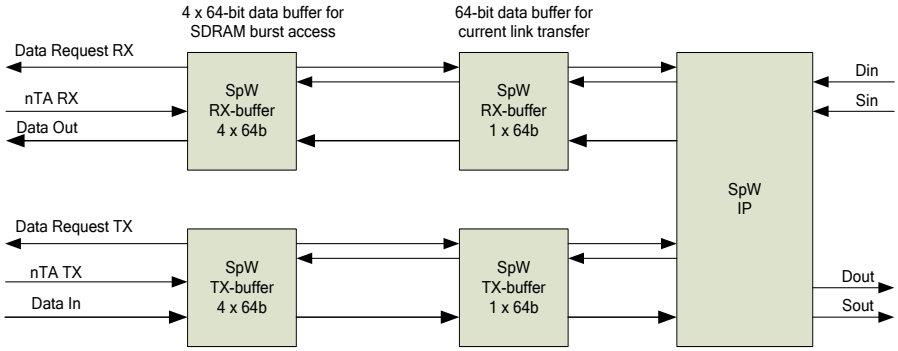


Figure 3 SpaceWire data buffers for memory burst access

2.2 DMA CHANNEL FOR RMAP

RMAP IP core has a common 32-bit bus interface for RMAP Initiator command encoder, Reply decoder and Target functions [7]. In the DPU, each RMAP core has access to SDRAM memory by the bus interface via it's own DMA channel, as is depicted in **Figure 4**. Memory transfer is one 32-bit word (4 bytes) per an access in the current version of the DPU. This leads to 29 Mbytes/s theoretical data write rate to SDRAM and 50 Mbytes/s data read rate for RMAP access.

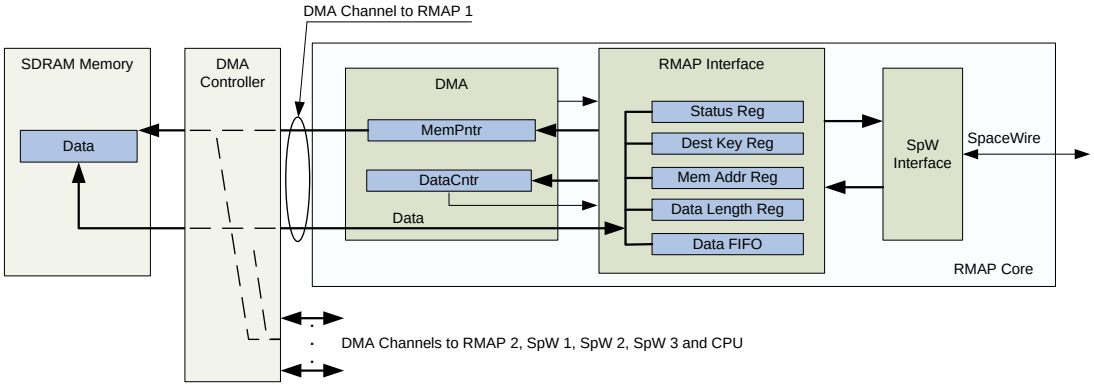


Figure 4 RMAP Core Interface to Memory

3 PERFORMANCE

3.1 CPU PERFORMANCE

CPU performance is highly dependent on whether code and data processed at the moment are located in the CPU cache or not. CPU can reach over 1000 MIPS performance during the code and data are available in the CPU cache. Cache fetches and flushes to memory are done via the 64-bit 60x bus, which operates at 50 MHz in the DPU. These operations may slow down performance considerably. On the other hand CPU has ability for branch prediction for more effective code fetch and SRAM and SDRAM interface functions effectively with burst accesses, 32 bytes are transferred per a burst. With PowerPC, burst accesses are performed when caches are enabled. In current version of the DPU, SDRAM memory is controlled by the SpWCU, which in turn is connected by a 32-bit, 50 MHz bus to the CPU.

Measured DPU power consumption is 10 W in SpaceWire loop-back test, and 8.5 W in idle. Power consumption is dependent on overall activity, i.e. activity in the CPU, SCU, SpWCU, memories and external interfaces.

3.2 SPACEWIRE PERFORMANCE

The three DPU SpaceWire links can connect and reach up to 100 Mbits/s link speed. Transmit speed is limited in the DPU to 100 Mbits/s, wherein receiver adapts to incoming link speed. Receivers have been tested successfully at 200 Mbits/s in a laboratory. Continuous 100 Mbits/s in the SpaceWire link results to 10 Mbytes/s data rate in one direction, and 60 Mbytes/s for three concurrent links full-duplex. The DPU DMA interface between SpaceWire and SDRAM memory is able to handle the rate.

Loop-back test in the laboratory measures continuous 10 Mbytes/s transfer rate between two active SpaceWire links and SDRAM. The CPU initiates every packet transmit and receive; the data rate depends on the software performance. This may explain the measured lower transfer rate than is expected.

3.3 RMAP PERFORMANCE

Every RMAP packet has a header, minimum length of 16 bytes for command and 12 bytes for reply, plus a CRC byte for data bytes in a packet [6]. Protocol overhead may

be thus large when a small amount of data bytes is transferred in a packet. However, when an RMAP command is started, CPU needs to write a transaction record and a header information record in the SDRAM memory for the becoming transaction to be read by the RMAP core [7], in addition to actual data transferred in a link.

RMAP loop-back test for variable size data packets (1 byte to 512 bytes) was arranged between two DPU boards, for which 4 Mbytes/s continuous RMAP – SDRAM transfer rate was measured. Initiator board generated packets and verified data correctness of replies. Target board simply wrote the received data in memory for a write command, and read data from memory and sent it in reply packet for a read command. It was not started a new command until the current reply was checked.

RMAP Target latency, 7.9 μs, was measured as time taken from command done to reply received status in the Initiator.

4 CURRENT AND FUTURE WORK

Tests to verify correct behaviour, data rates and reliability are performed for the DPU currently. Future work might include performance enhancement by integrating functions from the Mezzanine board in the DPU main board, which would allow Inter bus i.e. CPU access to SDRAM and RMAP speed up from current 200 Mbytes/s to 400 Mbytes/s. RMAP memory access data rate could be increased by exploiting SDRAM burst transfer mode as is done with the SpaceWire interfaces.

5 ACKNOWLEDGEMENTS

The authors acknowledge the support of ESA, University of Dundee, Astrium Ltd and Patria Aviation Oy for the work.

6 REFERENCES

1. Freescale Semiconductor, “MPC7448 RISC Microprocessor Hardware Specifications”, MPC7448EC, Rev. 4, 3/2007.

2. Freescale Semiconductor, “MPC7450 RISC Microprocessor Family Reference Manual”, MPC7450UM, Rev. 5 1/2005.

3. ECSS, “SpaceWire: Links, nodes, routers and networks”, ECSS-E50-12A, January 2003

4. ECSS, “SpaceWire- Remote memory access protocol”, ECSS-E-ST-50-52C, 5 February 2010.

5. Steve Parkes, Chris McClements, Martin Dunstan, “SpaceWire RMAP IP Core”, International SpaceWire Conference 2008, Nara Prefectural New public Hall, Japan, November 2008.

6. ECSS, “Remote memory access protocol”, ECSS-E-50-11 Draft F, December 2006.

7. Space Technology Centre, University of Dundee, “SpaceNet – RMAP IP Core User Manual Issue 1.5”, RMAP IP WP2-400.3, May 2009.

GR712RC – A MULTI-PROCESSOR DEVICE WITH SPACEWIRE INTERFACES

Session: SpaceWire Components

Short Paper

Sandi Habinc, Jiri Gaisler

Aeroflex Gaisler, Kungsgatan 12, SE-411 19 Göteborg, Sweden

sandi@gaisler.com jiri@gaisler.com

ABSTRACT

The GR712RC device is a first of its kind, offering the space community powerful multi-core processor capability in combination with multiple RMAP [6] enabled SpaceWire links [5]. The device is highly configurable and can operate in many different applications, ranging from platform control to payload processing.

GR712RC DUAL CORE LEON3-FT

1.1 OBJECTIVES

The GR712RC device has been designed to provide high processing power by including two LEON3FT 32-bit SPARC V8 processors [7], each with its own high-performance IEEE-754 compliant floating-point-unit and SPARC reference memory management unit. This high processing power is combined with a large number of serial interfaces, ranging from high-speed links for data transfers to low-speed control buses for commanding and status acquisition. The GR712RC can be utilized in symmetric or asymmetric multiprocessing mode. The processors provide hardware support for cache coherency, processor enumeration and interrupt steering.

1.2 FEATURES

The main functions of the GR712RC device are:

- 2 x LEON3FT 32-bit SPARC V8 Processor with
  - IEEE-754 High-performance Floating Point Unit (FPU)
  - SPARC V8 Reference Memory Management Unit (MMU)
  - 4x4kByte Instruction Cache, 4x4kByte Data Cache
  - Branch prediction
- Debug Support Unit with JTAG Debug Interface
- 6 x SpaceWire links, of which two with RMAP
- 10/100 Mbit/s Ethernet MAC
- Redundant Mil-Std-1553B BC/RT/MT (A/B)
- 2 x CAN 2.0B
- I^2C and SPI interfaces
- 8 x Timers, 6 x UARTs, Interrupt Controller, General Purpose Input/Output
- CCSDS/ECSS Telemetry and Telecommand [1][2][3][4]
- 192 kByte On-chip SRAM with ECC
- Memory controller for SRAM/PROM/SDRAM/IO with BCH and RS protection

The variation in interfaces allows different systems to be implemented using the same device type, which simplifies parts qualification and procurement. It also brings cost reductions to software development since the core functionality can be reused from application to application, only changing the drivers for the interfaces. Due to the high amount of peripherals and a limited number of pins there is an I/O switch matrix that controls which peripheral is connected to each pin.

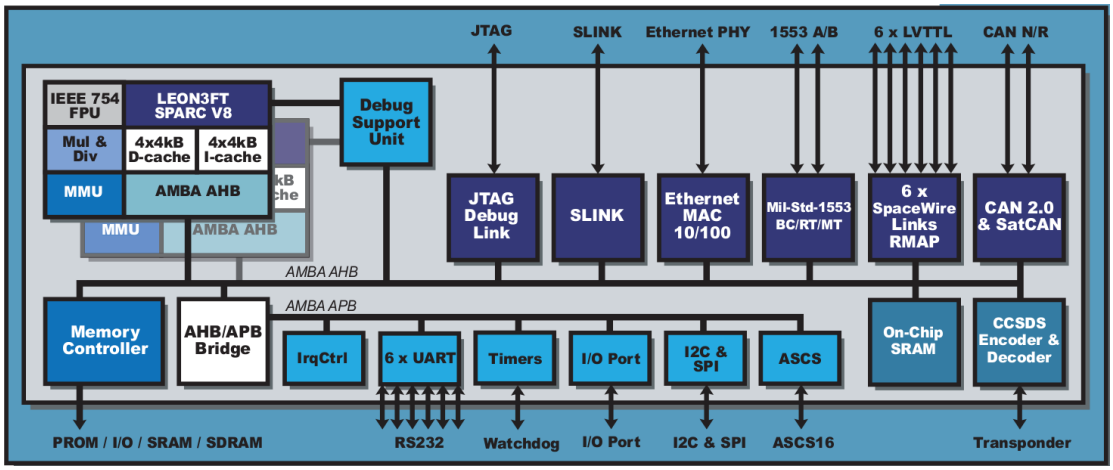


Figure: GR712RC block digram

The clock gating unit can turn off the clock for each major peripheral thus lowering power consumption considerably. The processor clock is automatically turned off when a processor is in power down mode. The FPU is clock gated when floating point operation is disabled or when the corresponding processor is powered down.

1.3 PERFORMANCE

The expected performance of the GR712RC device at 100 MHz system clock frequency is approximately 250 Dhrystone MIPS. The expected speed of the SpaceWire links is above 200 MBPS.

1.4 TECHNOLOGY

The GR712RC is fabricated at Tower Semiconductors Ltd., using standard 0.18 µm CMOS technology. It employs Rad-Hard-By-Design methods from Aeroflex Gaisler and Ramon Chips. As a preparation for this development, a first ASIC silicon prototype, the GR702RC, with integrated processor and SpaceWire interfaces has been successfully manufactured, validated and undergone radiation testing. The final GR712RC device is expected to be latch-up free, be fully protected against single event upsets in registers and memory, and tolerate a high total ionizing dose.

To allow flexibility and possibility for fault-containment, the Low-Voltage Differential Signaling (LVDS) buffers required for the SpaceWire interfaces need be implemented off-chip using standard components.

1.5 PACKAGING AND AVAILABILITY

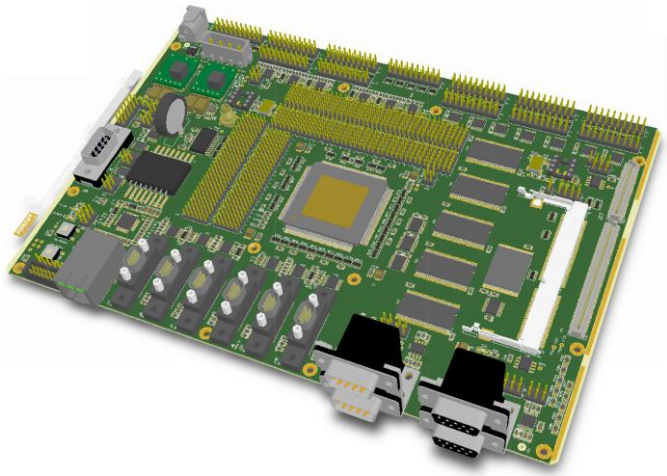
GR712RC will be packaged in a 240-pin Ceramic Quad Flat Package (CQFP) meeting high pin count, low die size, manufacturability and testability requirements.

The GR712RC will be available as engineering samples in August 2010. Space qualified parts will be available in first half of 2011. Prototyping and evaluation is possible using the GR712RC development board.

GR712RC DEVELOPMENT BOARD

In order to provide a platform for customers to begin developments using the GR712RC device, Aeroflex Gaisler provides a GR712RC development board. The board comprises a custom designed PCB in Compact PCI 6U format which can be used either stand alone or inserted into a CPCI rack.

The board has interfaces for all peripherals and 8 MByte of SRAM (with check-bits), 8 MByte of Flash PROM, and a standard SDRAM SODIMM socket. Each pin in the I/O switch matrix is configured with a jumper. All the interfaces are conveniently



located on the front side of the board, allowing easy access to a CPCI front-panel.

Figure: GR712RC development board

MULTIPROCESSOR NETWORKS OVER SPACEWIRE

The multiple SpaceWire links allow the GR712RC device to be used in multiprocessor applications, providing two high-performance processors in each node.

The onboard payload reference network is a marriage between the high-speed backbone SpaceWire network and the low-speed spacecraft control bus based on CAN or Mil-Std-1553B. The SpaceWire network is well suited for bulk data transfers, whereas the CAN or Mil-Std-1553B bus is suited for control and sensor acquisition tasks. The need to bridge the networks can be seen from several perspectives.

The GR712RC can be integrated in the Instrument Controller Unit (ICU) which acts as the payload data processor and mainly receives payload data from instruments and produces processed data to be downlinked. The main data communication is performed via the SpaceWire network. The ICU is however controlled and monitored via the CAN or Mil-Std-1553 bus from the On-Board Computer (OBC). The GR712RC then acts as a remote terminal which is being managed by the OBC.

Alternatively, the GR712RC can be integrated in the OBC. Since the OBC acts as the network manager on the CAN network, the CAN controller must carter capability such as node management and time distribution. The same approach applies to the Mil-Std-1553B bus. The OBC also communicates or manages the SpaceWire network via SpaceWire links.

As can be seen from the two above application scenarios, the capabilities of the GR712RC is not limited only to support the CAN/Mil-Std-1553B buses in an ICU, but will also allow its usage in the OBC. This will reduce development costs since the same device is used in both payload and avionics. This will also promote the usage of hybrid SpaceWire and CAN/Mil-Std-1553B networks. To support both applications, the GR712RC facilitate sufficient processing power as well as suitable interfaces.

To bring the concept a step even further, one could envisaged applications in which the GR712RC actually replaces the signal processor in an ICU or instrument. The processing capacity of the GR712RC then needs to be at least as great as can be expected for current monolithic signal processor devices. Such a concept would provide great savings in terms of power and board area, since a single device with some external memory would be sufficient to form the signal processing core.

The ultimate step would be to use the GR712RC processing capacity and number of SpaceWire links to take on the same role as the Transputer [9] used to have. This requires that the on-chip processor provides adequate floating point capacity. At the same time, the architecture is be scalable and allows operation under low power conditions by means of reducing the clock frequency and the use of few external memory devices. It is not clear if there will be a follow on for the current European digital signal processor, as 32-bit DSP manufacturers are not numerous and are in general not interested in licensing their technology for a niche sector like space. The GR712RC can fill this gap since being sufficiently performant.

Examples of how the GR712RC device can be used to build a fault-tolerant SpaceWire computer is discussed in [8].

CONCLUSIONS

The GR712RC device brings multi-processing to avionics and payload applications, increasing the processing performance compared to existing solutions, without consuming board real estate or demanding complex memory implementations.

The GR712RC development board has been designed to support initially stand alone operation, but also to fit into future architectures where inter-board communication is realized through active or passive SpaceWire backplanes.

REFERENCES

1. TM Space Data Link Protocol, CCSDS 132.0-B-1, www.ccdsds.org
2. TC Space Data Link Protocol, CCSDS 232.0-B-1
3. Telemetry transfer frame protocol, ECSS-E-50-03A, www.ecss.nl
4. Telecommand protocols, synchronization and channel coding, ECSS-E-50-04A
5. SpaceWire - Links, Nodes, Routers and Networks, ECSS-E-ST-50-12C

6. SpaceWire - Remote memory access protocol, ECSS-E-ST-50-52C
7. The SPARC Architecture Manual, Version 8, SPARC International Inc.
8. Ran Ginosar, *"A Fault-Tolerant SpaceWire Computer"*, ISC 2010
9. M.D. May et al., *"Networks, Routers and Transputers: Function, Performance, and Applications"*, SGS–THOMSON Microelectronics Group
10. Detailed GR712RC information: www.gaisler.com

SPACEFIBRE – HIGH SPEED FIBRE OPTIC DATA LINKS

Session: SpaceWire Components

Short Paper

Taisto Tuominen[(1)], Veli Heikkinen[(2)], Eveliina Juntunen[(2)], Mikko Karppinen[(2)], Kari Kautio[(2)], Jyrki Ollila[(2)], Aila Sitomaniemi[(2)], Antti Tanskanen[(2)], Mathias Pez[(3)], Norbert Venet[(4)], Iain McKenzie[(5)], Rory Casey[(6)], Demetrio Lopez[(7)]

[(1)]Patria Aviation Oy, Tampere, Finland
[(2)]VTT, Oulu, Finland
[(3)] D-Lightsys S.A.S, Rosny-sous-Bois Cedex, France
[(4)] Thales Alenia Space, Toulouse, France
[(5)] ESA/ESTEC, TEC-MME, Noordwijk, The Netherlands
[(6)] Fibrepulse Ltd., Co. Mayo, Ireland
[(7)] Alter Technology Group Spain Madrid, Spain

E-mail: taisto.tuominen@patria.fi, veli.heikkinen@vtt.fi, mathias.pez@d-lightsy.com, norbert.venet@thalesaleniaspace.com, iain.mckenzie@esa.int, rory@fibrepulse.com, demetrio.lopez@alter-spain.com

ABSTRACT

This paper describes development of two types of high performance fibre optic transmitter-receiver modules with optical fibre cables for SpaceFibre data links. Both solutions use mainly the same electrical and optoelectronic components but they have different mechanical designs. On type 1 transceiver electronics uses low temperature co-fired ceramic substrates (LTCC) and the module is hermetically sealed in Kovar housing. On type 2 an integrated ceramic package with fibre hermetic feed through has been developed in conjunction with a high performance optical sub assembly. On both types the transmitter is based on a 850nm GaAs Vertical Cavity Surface Emitting Lasers (VCSEL) and receiver part on a GaAs PIN photodiode.

1 INTRODUCTION

Emerging data rates on telecom satellites will require high through-put solutions for data transmission between e.g. antennas and data handling units. SpaceFibre is a proposed very high speed serial data link technology intended to complement the existing SpaceWire high-speed data link standard. This technique is based on opto-electronic transmitter and receiver modules with optical cables and connectors. Thus, two electronic units can be connected to each other with a high speed optical link.

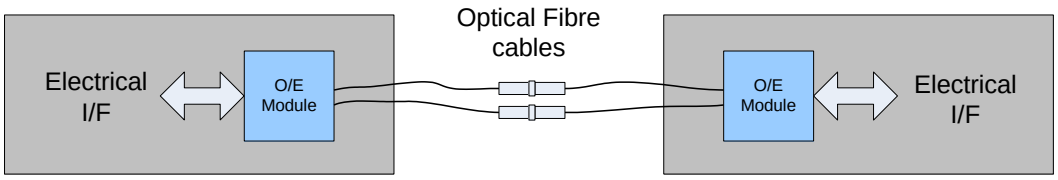


Fig. 1. Optical Link principle.

In the presented work we have developed two types of opto-electronic modules. VTT and D-Lightsys have both designed 10 Gbps optical transceivers suitable for the SpaceWire Protocol. For interoperability reasons, common mechanical dimensions, electrical pin layout and optical connector types were specified for the modules. The modules have max. outer dimensions of 17 x 17 x 5 mm^3 and is a 48-pin QFN type with 1mm pitch. A radiation resistant 50/125-µm 0.23NA graded-index fibre was selected for the transceiver pigtails. The outer diameter of the cabled pigtail is 1.2 mm and it is compatible with the Diamond Mini-AVIM, Diamond standard AVIM and Radiall LuxCis connectors.

2 MODULE DESIGNS

2.1 VTT'S TRANSCEIVERS

VTT employs low-temperature co-fired ceramic (LTCC) technology for the transceiver module electronics. Low conductor resistance and dielectric loss, multilayer structures with fine-line capability and compatibility with hermetic sealing make LTCC a useful technology for high-speed data communications. In addition, the good match of the thermal expansion coefficient to optoelectronic chips reduces packaging-induced thermomechanical stresses.

We decided to use 850-nm vertical cavity surface emitting lasers (VCSEL) as the light emitters. They offer low drive current and small power consumption. GaAs PIN diodes were chosen for the photodetectors. VCSELs and PIN diodes are bare dies compatible with flip-chip bonding.

Transceiver electronics is based on commercial 10 Gbps components and it consists of three boards: the mother board, transmitter optical subassembly (TOSA) and receiver optical subassembly (ROSA). The mother board contains the laser driver and interconnections between the subassemblies and current mode logic data inputs/outputs. TOSA contains the VCSEL and few passive components, and ROSA consists of the photo diode, receiver amplifier and few passives. The transceiver uses a single 3.3-V power supply and has a typical power consumption of 230 mW.

The maximum average transmitted optical power is limited to –0.67 dBm because of the eye safety limitations. The nominal sensitivity of the receiver at the photo detector is –18 dBm for 10^{-12} BER at 10 Gbps. Transceiver modules are hermetically sealed. This is realised with a metal lid and a Kovar frame soldered to the mother board, Fig. 2. The hermetic fibre feed-through is made using a low temperature glass preform. The transceiver module has dimensions of 17 × 17 × 5 mm^3 and a mass excluding the fibre pigtails of 4 grams. The pigtail cable weights 2.5 grams/meter.

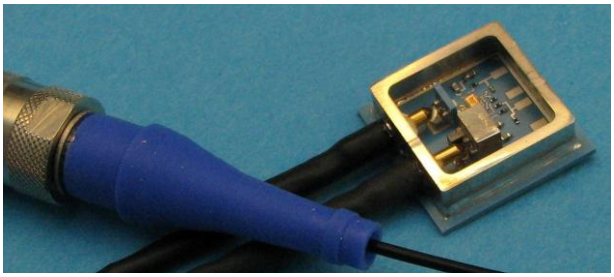


Fig. 2. VTT's SpaceFibre transceiver with LuxCis connectors before lid sealing.

2.2 D-LIGHTSYS'S TRANSCEIVERS

The module designed by D-Lightsys is realised with a HTCC ceramic substrate with a KOVAR wall and a low temperature glass feedthrough.

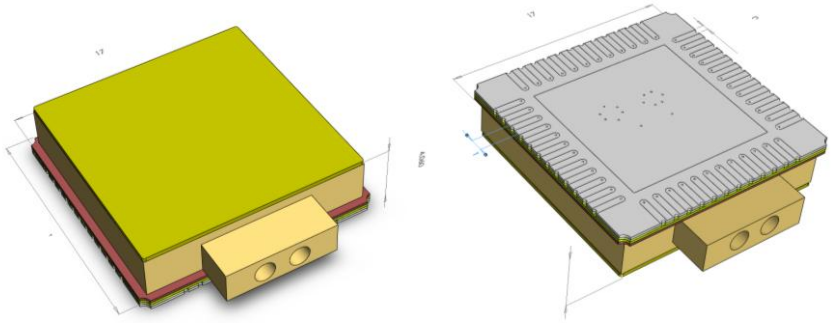


*FIG. 3 - HERMETIC CERAMIC PACKAGE (FRONT AND BOTTOM VIEW)*REFERENCES

D-Lightsys develop a unique Planar Optical Sub-Assembly (OSA) technology based on silicon micro machining. This silicon bench is used to precisely align the fibres (with +/-5µm accuracy) in front of the laser source and the photo detector chip to guarantee a coupling efficiency better than 80 to 90%.

The Optical Sub Assembly, is less than 2mm thick and have dimension of 6x6mm, allowing the integration of several laser source and detector manufacturers. The OSA is also responsible for the heat transfer within the package to allow the module to operate in the [-40;+85°C] temperature range.

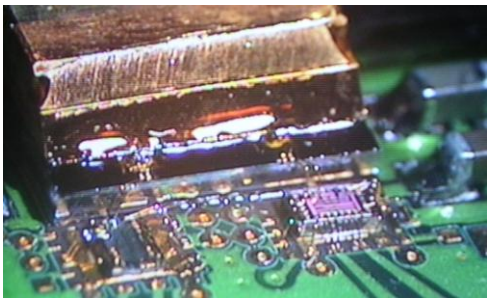


FIG. 4 – OPTICAL SUB-ASSEMBLY AND HIGH SPEED ELECTRONICS OVERVIEW

Laser driver and photodiode transimpedance/limiter amplifiers are hybridized within the hermetically sealed package with a controller to realize a protocol independent; temperature compensated high performance optical transceiver.

The module is designed to operate up to 10Gbps, with a 12dB link budget over the temperature range. D-Lightsys controller algorithm monitor in real-time the module status and temperature and compensate the laser modulation and biasing current accordingly. Less than 1dB of average optical power and less than 2dB of extinction ratio variation over temperature could be achieved.

3 PRELIMINARY TEST RESULTS

3.1 VTT'S MODULE : PRELIMINARY TEST RESULTS

Only the hermeticity test was performed for VTT modules at the time of issuing this paper. The results were positive and other test results were expected to be available soon after.

3.2 D-LIGHTSYS'S MODULE : PRELIMINARY TEST RESULTS

The following graph plots the evolution of the transceiver performances over the temperature range for bit rate of operation of 7Gbps.

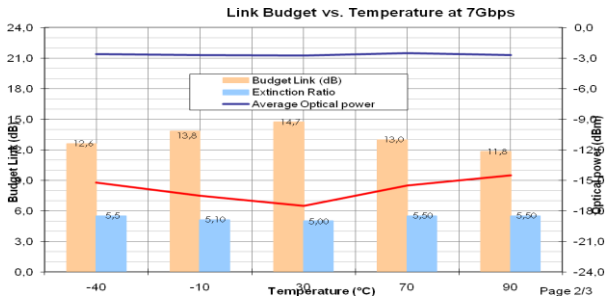


FIG. 5 – PRELIMINARY TEST RESULTS AND OPTICAL EYE DIAGRAM AT 90°C

A link budget better than 12dB could be achieved over the temperature range. The receiver sensitivity is slightly impacted with the temperature due to the test board substrate limitations (losses, reflexions, etc...). We provide hereafter the eye diagrams done a 90°C for the transmitter (optical eye) and -10°C for the receiver (Electrical Eye).

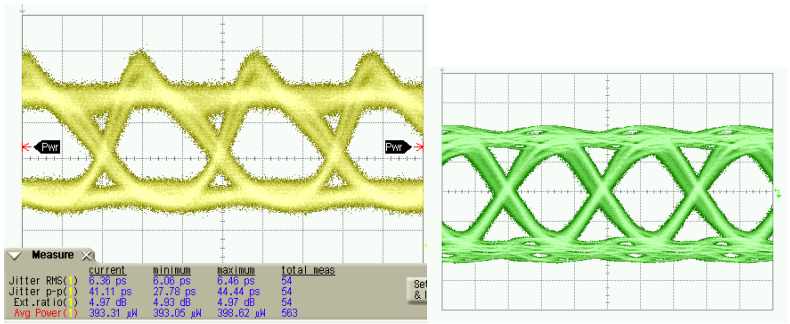


FIG. 6 – ELECTRICAL AND OPTICAL EYE DIAGRAM AT 7GBPS

3.3 RADIATION TESTS FOR COMPONENTS OF BOTH DESIGNS

Heavy Ion, protons and Gamma radiation tests have been performed within the scope of the project. Modules show good behaviour with a total Gamma dose of 100kRad and with proton radiation up to 1E12 protons/cm at a rate of 2E8 protons/cm2/s.

At Heavy Ions test up to LET threshold of 70 MeV/mg/cm2 only the VCSEL driver circuit showed to be latch up sensitive. Means to overcome this problem are being studied with the support of the components manufacturer.

4 CONCLUSION

Test results available so far are promising but more detailed testing will be required full characterisation of the developed optoelectronic modules.

SPACEWIRE BACKPLANE WITH HIGH-SPEED SPACEFIBRE LINK

Session: SpaceWire Components

Short Paper

Minoru Nakamura, Tatsuya Ito, Yasutaka Takeda

Advanced Technology R&D Center, Mitsubishi Electric Corp., 8-1-1, Tsukaguchi-Honmachi, Amagasaki, Hyogo, 661-8661, Japan

Isao Odagi, Shinya Hirakuri

Kamakura Works, Mitsubishi Electric Corp., 325 Kamimachiya Kamakura, Kanagawa 247-8520 Japan

Keitaro Yamagishi, Koji Shibuya

Information Technology R&D Center, Mitsubishi Electric Corp., 5-1-1 Ofuna, Kamakura, Kanagawa 247-8501, Japan

Masaharu Nomachi

Laboratory of Nuclear Studies, Graduate School of Science, Osaka University

1-1 Machikaneyama, Toyonaka, Osaka 560-0043, Japan

E-mail: {Minoru.Nakamura@ea, Ito.Tatsuya@ak, Takeda.Yasutaka@aj, Odagi.Isao@cb, Hirakuri.Shinya@da, Yamagishi.Keitaro@bk, Shibuya.Koji@ak}.MitsubishiElectric.co.jp, nomachi@lns.sci.osaka-u.ac.jp

ABSTRACT

SpaceWire Backplane is a backplane architecture with SpaceWire interconnects between each board. In ISC2008, we proposed that the space qualified, non-high speed backplane connector evaluated and, if the signal assignment is carefully designed, can be used for high-speed data transmission.[1]. On the basis of this result, we developed the SpaceWire backplane prototype with SpaceWire and SpaceFibre links. This backplane prototype includes a passive backplane board with SpaceFibre links of 3 Gbps or higher. We also developed a sample daughter board for functional evaluation that has both SpaceWire and SpaceFibre interfaces. We present the design and implementation of the SpaceWire backplane as well as an evaluation of the high-speed signal quality of the backplane and daughter board prototyping result that includes the SpaceFibre interface implemented.

1 BACKPLANE TOPOLOGY AND SIGNAL ASSIGNMENT

In the backplane interconnection system, high-speed signals are connected between daughter boards via connecters and a backplane. This reduces the number of harnesses needed and improves the integrity of the signal. SpaceWire backplane is a backplane architecture that uses SpaceWire as an interconnect between boards. While SpaceWire is point-to-point link connection, backplane topology and/or router functionality is important. We decided our backplane is passive without any active components like routes because the backplane needs to be reliable and durable. Although there are

various network topologies such as tree, ring and mesh. we selected a backplane configuration as the full mesh topology for the followings reasons.

1. The backplane has to have enough flexibility. While other topologies are a subset of the full mesh topology, the actual configuration can be defined by each satellite and/or purpose of each component.

2. Usually full mesh topology needs too many interfaces to construct a network system. However, almost all components of a spacecraft can be constructed with no more than 8 boards. Therefore backplane needs to have only 7 SpaceWire ports.

3. Full mesh topology can be partitioned in any subset. This nature is useful to make a component redundant. We defined two partitions of 4 slots to our backplane and defined the power supply board connector and control signals to each partition. If full of 8 slots required for 1 component, redundancy can be provided by using 2 chassies connected to each other via external SpaceWire signals.

4. Daughter boards can be independent of slot position if SpaceWire ports are assigned in cyclic symmetrically. The daughter board thus needs to implement only the required SpaceWire ports. For example, for a backplane partitioned in two 4 slots, each daughter board need only the first three SpaceWire ports.

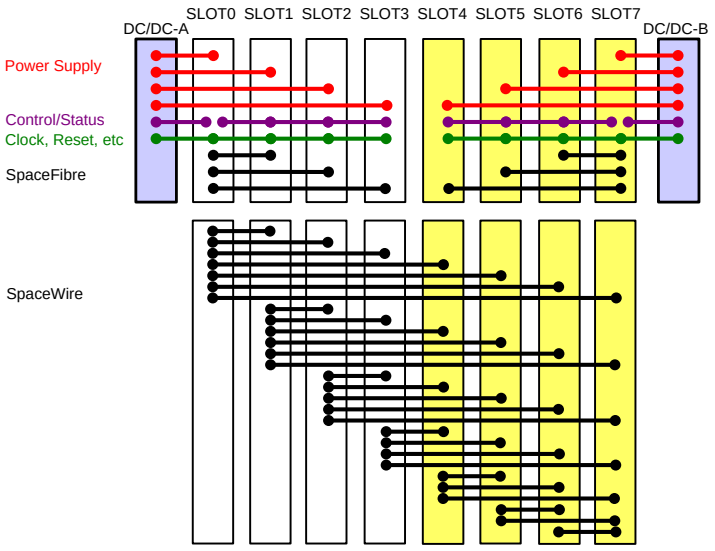


Figure 1 : SpaceWire Backplane topology

On the contrary, we defined Tree topology for SpaceFibre channels because fewer SpaceFibre channels can be assigned to one connector with high-speed signal quality. We suppose the purpose of SpaceFibre channel is for high speed/large capacity data recorder and/or high-speed signal processing boards, so the tree topology is suitable for this purpose. The backplane topology we defined is shown in Figure 1.

The definition of control/status signals is also important as it defines the topology of the SpaceWire topology. We defined clock, timing pulse distribution, power control/status, slot ID and reset signals. We also defined slot0 and slot7 as control master slot of other slots. The control master slot controls power on and off via power control/status signals.

We also defined mechanical design including chases and daughter board. As the mechanical design is based on our existing components, using this prototype to develop the flight model is rather easy. The daughter board is designed with upper compatibility to a 6U Euro card. The circuits designed for this card are easily implemented onto the daughter board because the circuit-populated area excluding frame structure is same as that of the 6U Euro card. The prototype specification is shown in Table 1.

Number of slots	8 + 2(DC/DC)
Number of channel	SpaceWire : 7ch for each slot
	SpaceFibre : 3ch for slot0 and slot7, 1ch for other slots
Topology	Full mesh(SpaceWire), Tree(SpaceFibre)
Dimensions	Chassis : 290 mm x 253 mm x 264 mm
	Daughter board : 265 mm x 223 mm
Backplane connector	110 pins x 2 / slots

Table 1 : Prototype backplane and chassis specifications

We developed and evaluated SpaceWire backplane prototype. While we presented 3Gbps signal integrity in ISC2008[1], we designed this prototype to transmit signals up to 6.25Gbps because SpaceFibre maximum transfer rate target is 6Gbps. To achieve this high-speed signal capability, we also designed an equalizer for high-speed signal lines. The daughter board with the SpaceWire and SpaceFibre interfaces is also developed to validate functionality of the backplane system. In addition to the SpaceWire codec we have already developed[2], SpaceFibre codec is developed to this daughter board. A block diagram of the daughter board we developed is Figure 2. In addition to backplane connector signals, two SpaceWire and two SpaceFibre connectors are assigned in front. All SpaceWire and SpaceFibre codec channels, and a small processor for debug and port control and switch fabric are integrated into FPGA. The first implementation of SpaceFibre codec works in loopback mode. We are planning an interoperability test with other SpaceFibre implementations to validate and improve the SpaceFibre specification.

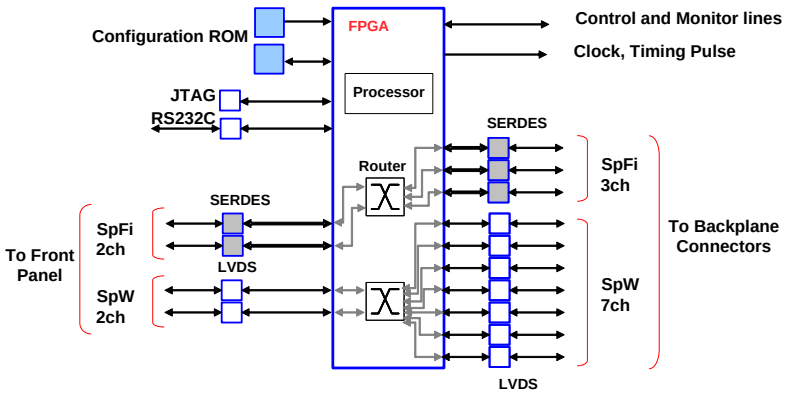


Figure 2 : Daughter board block diagram

The backplane board, the daughter board and the chassis we developed are shown in Figure 3 and the measurement of transmission characteristics and eye pattern of 6.25Gbps backplane signal are shown in Figure 4. Our equalizer significantly

improves both transmission characteristics and eye pattern. This result shows our backplane design transmits data at 6.25 Gbps.

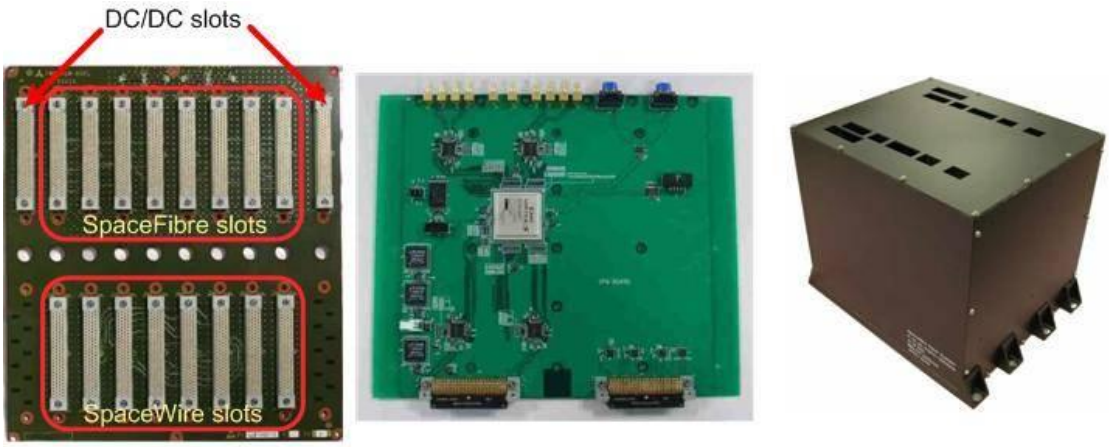


Figure 3 : SpaceWire backplane (left), daughter board (center), and chassis (right)

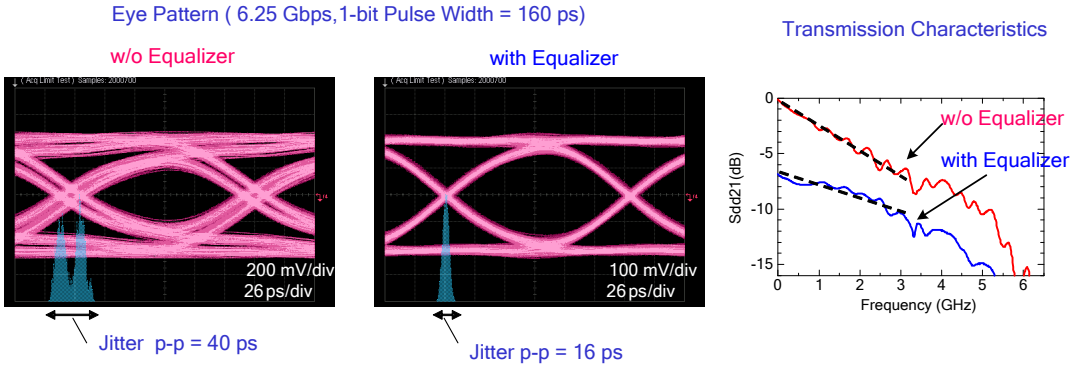


Figure 4 : Evaluation of high-speed signal transmission

2 CONCLUSION

We defined SpaceWire backplane architecture including network topology, signal assignment, and mechanical design. We then developed the prototype of backplane and daughter board with a chassis to evaluate our backplane system. The evaluation result of the backplane signal integrity is enough to transmit a 6 Gbps high-speed signal via a space-qualified connector. We also implemented SpaceFibre codec for our daughter board. The first implementation works with a loopback test and we are planning to test interoperability with other implementations.

3 REFERENCES

1. Shibuya, Yamagishi, Oh-hashi, Saito, Nomachi, “Evaluation and Analysis of Connector Performance for the SpaceWire Back Plane”, International SpaceWire Conference 2008, Nov.2008, pp 83-86.

2. Sasaki, Nakamura, Yoshimoto, Yoshida, Yoshikawa, “A CPU Module for a Spacecraft Controller with High Throughput SpaceWire Interface”, International Conference SpaceWire Conference 2008, Nov.2008, pp 181-184

OPTIMIZATION OF REDUNDANT ANALYSIS AND DESIGN FOR SPACEWIRE

Session: SpaceWire Components

Short Paper

Yong GUAN, Guohui WANG

College of Information Engineering, Capital Normal University, Beijing, China

Jie ZHANG

College of Information Science & Technology, Beijing University of Chemical Technology, China

Shengzhen JIN, Yuanyuan SHANG, Huizhuo NIU

College of Information Engineering, Capital Normal University, Beijing, China

E-mail: gxy169@sina.com, wgh_boy@hotmail.com, zhang@mail.buct.edu.cn, jsz0622@163.com, syy@bao.ac.cn, ymdynhz@126.com

ABSTRACT

With limited resources available to acquire FPGA chips, a design method was developed based on the redundancy backup of the various functional sub-modules in SpaceWire node. The aims to find the optimization design of redundancy backup in an FPGA chip. This design method can improve the reliability of the whole system.

First, according to a failure rate of λ, the reliability of the control, transmitter, receiver, recovery, faulting, timing, reliability, and baud rate selection modules of the whole system was calculated, attaining a total of 0.8675. Second, to meet the reliability requirement of the whole system of the space-solar telescope for SpaceWire (0.95 after working in the 750 km sun-synchronous-orbit for three years), the reliability targets meeting the requirements of every module mentioned above were calculated and attained according to grading distribution. Finally, using the optimal solution based on linear programming in the theory which addressed the "optimal allocation of spare parts" problem, the optimization design of the redundancy backup of the various functional sub-modules was obtained. The simulation results proved the validity of this design, indicating that a new method in improving the reliability of the redundancy backup in the SpaceWire system has been developed.

1 INTRODUCTION

In meeting modern product quality requirements, reliability is of prominent importance. Backup program design and optimization is a basic work in reliability engineering. The SpaceWire designed should reach the target such that the reliability of the whole SpaceWire system is 0.95 after working in a 750 km sun-synchronous-orbit for three years. The initial design in FPGA has achieved a reliability of 0.8674, which is far from what was expected. However, in order to realize the SpaceWire node in an FPGA chip and perform scientific backup, high reliability requirements must be met, and the limited resources on the FPGA chip must be maximized. Therefore, a design is proposed which perform each functional sub-module within an FPGA-based Spacewire node redundancy backup, thereby enhancing the reliability of the entire system to meet task demands.

2 Algorithm of the Backup

2.1 Reliability Calculation of The Sub-module

With the development of EDA technology, the use of hardware description language design has become a trend. The SpaceWire node is achieved on a Xilinx's Spartan-3E (XC3S250E) FPGA, and the sub-module-level backup of the system is completed at the same time. To obtain the reliability of a sub-module before backup, the following are assumed: Ⅰ. A blank FPGA's reliability is 1. Ⅱ. The probability of Single Event Upset, electromagnetic interference, and other incidents is equal within the duration of three years. Ⅲ.The probability of damage in the working CLB of FPGA is the same. Ⅳ. The reliability of CLB which does not work is 1, and CLB is not damaged.

Let R_i be the reliability of the sub-module; let i=1,2,…,8. Let AP_i be the occupied area ratio for each sub-module; let i=1,2,…,8. If the occupied area ratio of the sub-module is small, then the effect is lesser, and the normal working hour is long. For the opposite condition, the effect is greater, and the normal working hour is short. The FPGA we used, whose CLB distribution is shown in Table 1. The number of CLBs occupied by each sub-module in the SpaceWire node is shown in Table 2. Let A_i be a factor of influence of the area, for i=1,2,…,8. Then the formula is

$$A_i=\frac{1}{AP_i} \tag{1}$$

Let t_0 express the normal working hour, then the Mean Time to Failure is

$$MTTF_i = A_i \times t_0 \tag{2}$$

The system failure rate for each sub-module is

$$MTTF_i = \int_0^{\infty} e^{-\lambda t} dt = 1/\lambda \tag{3}$$

Thus, the obtained formula for calculating the reliability of each sub-module is

$$R_i(t) = e^{-\lambda_i t_0} \tag{4}$$

Based on above Formulas (1), (2), (3), and (4), the reliability of the 8 sub-modules calculated is shown in Table 2. Then according to the series system reliability formula, the entire system reliability achieved is R = 0.8674.

Table 1: Spartan-3E FPGA Data Sheet[1]

IC	CLB		
	Row	Arrange	Total
XC3S250E	34	26	612

Table 2: The reliability of each sub-module

Sub-module	CLBs	Reliability
control module	12	0.9806
sending module	20	0.9678
receiver module	32	0.9491
recovery	12	0.9806
faulting module	2	0.9967
timing module	5	0.9919
credibility	2	0.9967
baud rate selection	2	0.9967

2.2 Rating Distribution Method

The rating distribution method[2] is determined by peer experts based on the complexity, importance, and degree of difficulty to achieve the technology, among others. Evaluation is then made based on these factors, which not only reflects the characteristics of the whole system but also underlines the recommendations from the long-term accumulated experience of experts in technology. Finally, an effective backup program is given. The rate of each sub-module is given by

experts according to the effect of the reliability of the system or the basic reliability of several key factors. Let K be the derived weighting factors for each sub-module, then the reliability is distributed.

Weighting the product depends on a variety of factors, including the characteristics of FPGA, as well as complexity, importance, hours of work, and technological factors. Such factors can be further divided into several levels. Generally, score is divided into five levels—*perfect, very good, quite good, good, and general*. According to the characteristics of the SpaceWire node, the experts present the results of the specific rates in Table 3.

Calculation of the reliability index for each sub-module:

Ⅰ. The weighing factors of the sub-module are

$$K_i = \frac{C_i}{C_s} \tag{5}$$

Ⅱ. The rating of sub-module is

$$C_i = \prod_{j=1}^{p} m_{ij} \tag{6}$$

Ⅲ.The rating of the system is

$$C_s = \sum_{i=1}^{n} C_i \tag{7}$$

Ⅳ. The sub-module reliability index is

$$\lambda_i = K_i \lambda_s \tag{8}$$

If the reliability index is R_s=0.95, then the system failure rate is λ_s=5.4217E-10 is calculated according to the formula ($R_s = e^{-\lambda_s t}$). The reliability distribution is completed depending on the formula ($R_i = e^{-\lambda_i t}$). The scores given by several experts in related fields are averaged, obtaining the result of each module by the formula (5), (6), and (7), as presented in Table 3. At the same time, the calculating of the specific circumstances of the reliability distribution through Formula (8) is also shown in Table 3.

Table 3: Index of each sub-module

Weighing factor	Complexity	Importance	working time	Technology	Rating of sub-module	Weighing factor	Failure rate	Reliability
control	3	2	2	5	60	0. 0866	0. 4694E−10	0. 9956
sending	5	1	4	5	100	0. 1443	0. 7823E−10	0. 9926
receiver	5	1	4	5	100	0. 1443	0. 7823E−10	0. 9926
recovery	3	2	4	5	120	0. 1732	0. 9388E−10	0. 9912
faulting	1	5	3	5	60	0. 0866	0. 4694E−10	0. 9956
timing	1	5	5	4	125	0. 1804	0. 9779E−10	0. 9908
credibility	1	4	4	5	80	0. 1154	0. 6259E−10	0. 9941
baud rate selection	1	4	3	4	48	0. 0693	0. 3755E−10	0. 9965
The Whole System					693	1	5. 4217e−10	0. 95

2.3 The Linear Programming Method

From Section 2.2, we learned that the rating distribution method is an effective backup plan, as pointed by the long-term technical background of experts. However, this method shares certain deficiencies like the subjective factor of experts, fuzzy weighted assignment and so on. The linear programming approach[3] is adopted. The relatively superior backup program can be discovered using the main factor for searching the most superior result. The scientific nature and validity of the allocation of the backup part are analytic, representing the complexity parameter aspect.

Let the system be composed of n inter-independent sub-modules. Every sub-module is realized by the same configurable logic block; let j=1，2，…，n,

C_j -- the area of sub-module ***j***.

K_j -- the number of sub-module ***j***.

$R_j(K_j)$--when the number of the ***j*** sub-module is K_j, the reliability of the sub-system ***j***.

Obviously, the reliability of the system is as follows:

$$R(k)=R(K_1，\ ...，\ K_n)=\prod_{j=1}^{n} R_j(K_j) \tag{9}$$

Let the system be composed of n inter-independent sub-modules. If the sub-module ***j*** is composed of K_j inter-independent modules ***j*** by parallel, let P_j, for $j = 1,2, \ldots, n$ be the reliability of modules ***j*** , and the reliability of the modules ***j*** by parallel is as follows:

$$R_j(K_j) = 1 - (1 - P_j)^{K_j} \tag{10}$$

Here, the reliability of the system is as follows:

$$R(k) = \prod_{j=1}^{n} R_j(K_j) = \prod_{j=1}^{n} [1 - (1 - P_j)^{K_j}] \tag{11}$$

Then the value of $K_1，\ \ldots，\ K_n$ must meet the following equation set:

$$\begin{cases} \prod_{j=1}^{n} R_j(K_j) \geq R_0 & (12) \\ K_j = 1,2, \ldots，\ j = 1,2, \ldots n & (13) \\ \sum_{j=1}^{n} C_j K_j = \min(\text{aire}) & (14) \\ \sum_{j=1}^{n} C_j K_j \leq 612 & (15) \end{cases}$$

Since the reliability of the system must meet the requirement, $R_s = 0.95$, the total number of parts of the system is considered least.

According to series systems by n-parts, equality is seen in the following: $R(1,1,\ldots,1)=p_1 p_2 \ldots p_n$。The same module is added to some sub-modules every time, until the reliability of the system is increased most quickly up to or exceeding R_s. Obviously, when the same module is added to the module in a parallel system in a situation where the reliability is $p_1，\ p_2，\ \ldots，\ p_n$ is smallest. According to the formula (12), (13), (14), and (15), the searching algorithm is as follows:

Algorithm 3.1 Using linear programming method to find the optimal solution. (MATLAB)

```
01: Input (Ri, Rs, Rp)
02: R=prod(Ri)
03: Te=Ri
04: while  R<Rs
05:  [T,j]=min(Te)
06:         Te(j)=1-(1-Ri(j)).^(Rp(j)+1)
07:         Rp(j)=Rp(j)+1
08:         R=prod(Te)
09: end while
10: Ri=Te
11: Output (R，Ri，Rp)
```

Ri--The reliability of each sub-module. **R_s----The target reliability of system.**

Rp--The initial number of each sub-module. **R-- System final reliability**

3 EXPERIMENTAL RESULTS

Tables 4 show the backup programs obtained by the rating distribution method and the linear programming optimization.

Table 4: The optimizing backup program

Sub-module	Reliability (Initial)	Program 1		Program 2		Program 3	
		program	backup	program	backup	program	backup
control	0.9806	2	0.9996	2	0.9996	1	0.9806
sending	0.9678	2	0.9990	2	0.9990	2	0.9990
receiver	0.9491	2	0.9974	2	0.9974	2	0.9974
recovery	0.9806	2	0.9996	1	0.9806	2	0.9996
faulting	0.9967	1	0.9967	1	0.9967	1	0.9967
timing	0.9919	1	0.9919	1	0.9919	1	0.9919
credibility	0.9967	1	0.9967	1	0.9967	1	0.9967
baud rate selection	0.9967	1	0.9967	1	0.9967	1	0.9967
System	0.8674		0.9778		0.9592		0.9592

Note: The number of occupied CLBs is 163 in program 1; The number of occupied CLBs is 151 in program 2 and 3

Tables 4 show the results. The rating distribution method can be given an effective backup program of the SpaceWire node, which includes the backup control, sending, receiver, and recovery module. The two integrated programs given by the linear programming method also arrive at the same conclusion. The kinds of backup solution used can make the overall system reliability reach 0.9778. The number of occupied CLBs is 163. Obviously, when the reliability meets the requirements, the on-chip resources of FPGA are saved as compared to the backup overall SpaceWire node. Therefore, the goal of optimized redundant backup is realized by this method.

4 CONCLUSION

In this paper, we develop an effective program of SpaceWire node backup, which is calculated by the rating distribution method. This program is also achieved by the classical mathematical theory named the linear programming method. This analytic method represents the scientific rating distribution method on the complexity factor. In short, in order to combine the two methods, the reliability requirements of the backup design must first be optimized. Therefore, the design and optimization algorithm we provide plays an important role, which has practical significance in redundancy backup research.

The next step focuses on using classical mathematical theory to represent other rating factors of the rating distribution method analytically, as well as scientific and reasonable for factors such as the importance scale, working time, technical, etc. Thus, in theory, the rating distribution method is improved.

5 REFERENCES

1. Xilinx, “Spartan-3E FPGA Family : Complete Data Sheet”, Xilinx, March 21, 2005

2. Liangqiao Li, Yaohua Xu, Shaofeng Dong, “Reliability technology and management”, Industry Press , 1991, p.150-155

3. Jinhua Cao, Pei Cheng, “Mathematical theory of reliability cited”, Higher education press China, 2006, p.37-76

4. Weibo Er, Xuchang Li, Weiqun Yang, "Based on Reliability Demand Model Analysis for Spare Parts of the Weapon System", Journal of Missiles and Guidance, vol27 No.1, 2007, p.332-336

RACE CONDITION FREE SPACEWIRE DECODER FOR FPGA

Session: SpaceWire Components

Short Paper

Masaharu Nomachi

Laboratory of Nuclear Studies, Graduate School of Science, Osaka University

1-1 Machikaneyama, Toyonaka, Osaka 560-0043, Japan

Shigeru Ishii, Yoshikatsu Kuroda, Kazunori Masukawa

Nagoya Guidance & Propulsion Systems Works, Mitsubishi Heavy Industries LTD

1200, O-aza Higashi Tanaka, Komaki, Aichi, 485-8561, Japan

E-mail: nomachi@lns.sci.osaka-u.ac.jp, shigeru_ishii@mhi.co.jp, yoshikatsu_kuroda@mhi.co.jp, kazunori_masukawa@mhi.co.jp

ABSTRACT

Data-Strobe encoding is a simple encoding system. A clock signal is simply recovered by the exclusive OR of Data and Strobe. However, this simple decoder causes race condition. When Data line changes the logic level, the recovered clock also changes the logic level. Therefore, the recovered clock and the Data are in race condition. Race condition obstructs portability of the decoder logic in FPGA.

We have developed the DS decoder to avoid race condition, which works up to the maximum speed of FPGAs. The race condition free decoder is portable to any FPGAs.

1 INTRODUCTION

Data-Strobe encoding is easy to be decoded with small amount of logic in a FPGA. Recovered clock by the exclusive OR of Data signal and Strobe signal can be used for decoder logic (Figure 1). However, this simple way causes race condition. Data signal and recovered clock may change at the same moment (Figure 2). Therefore, the recovered clock and the Data signal are in race condition. The behaviour of this simple logic depends on propagation delay and set-up/hold time of the FPGA. They depend on the

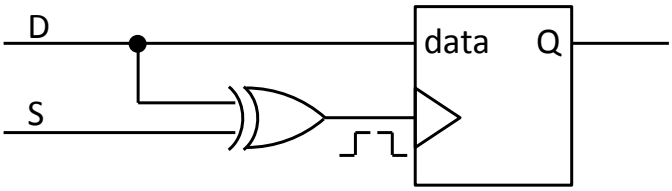


Figure 1: DS decoder logic

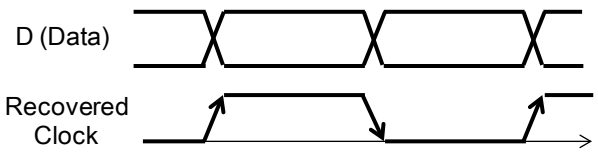


Figure 2: Wave form of Data signal and recovered clock.

routing in the FPGA chip. Thus, race condition obstructs portability of the decoder logic in FPGA. It should be noted that race condition stay also in slower bit rate.

Simple solution to avoid race condition is to synchronize the input signals. It is safe and simple way. However, the sampling interval must be shorter than the bit duration. Therefore, this kind of decoder cannot extract maximum performance of the FPGA compare to the simple decoder logic which uses the recovered clock. In this paper, we present new idea to avoid race condition, which works up to the maximum performance of FPGAs.

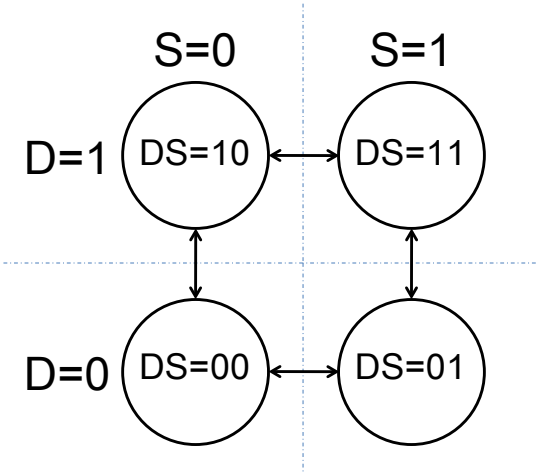


Figure 3: State diagram of Data (D) and Strobe (S).

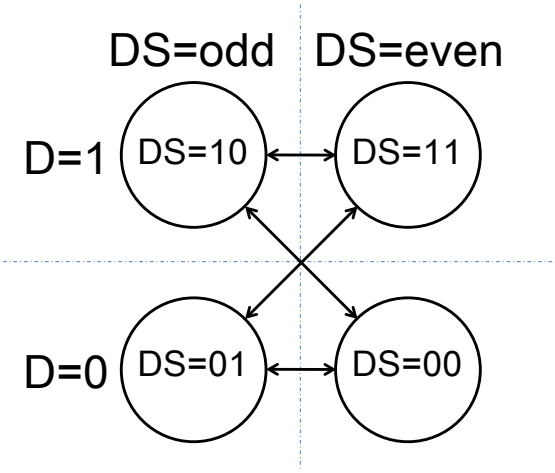


Figure 4: State diagram of D-S signals.

2 STATE DIAGRAM OF INPUT SIGNALS

We examine the behaviour of D (Data) and S (Strobe) in DS encoding. The figure 3 is a state diagram of D and S. There are no transitions between orthogonal states, which are caused by simultaneous change in both D and S. In DS encoding, simultaneous change of D and S does not happen.

The figure 4 also shows the state diagram but lower two states are twisted. States are classified by “parity” of “DS”. DS odd state is DS=”10” or DS=”01”. DS even state is DS=”11” or DS=”00”. In other words, exclusive OR of D and S is “1” in DS odd states and “0” in DS even states. DS odd states and DS even states appear alternatively (Figure 5).

The appearance rate of even/odd states is half of the bit rate. DS even state starts at the falling edge of the recovered clock. DS odd state starts at the rising edge of the recovered clock. Therefore it is possible to extend those states until the next occurrence, which is at the two bit after.

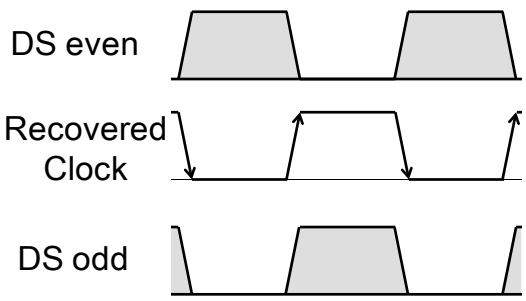


Figure 5: Wave form of DS even state and odd states

3 RACE CONDITION FREE LOGIC

Dividing the input states into two groups (even and odd), the frequency of the state in each group is a half of the bit rate (Figure 6). Extended even state changes the state at the falling edge of the recovered clock but does not change the state at the rising edge. Therefore, extended even state and the rising edge of the recovered clock are not at race condition. Extended odd state changes the state at the rising edge of the recovered clock but does not change the state at the falling edge. Therefore, extended odd state and the falling edge of the recovered clock are not at race condition.

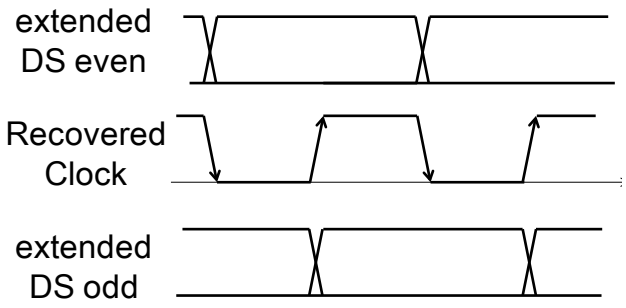


Figure 6: State diagram of D-S signals.

We split the state in to two groups, which are DS even states and DS odd states. The state is extended using asynchronous set / reset (Figure 7). The even/odd state is held for two bit duration.

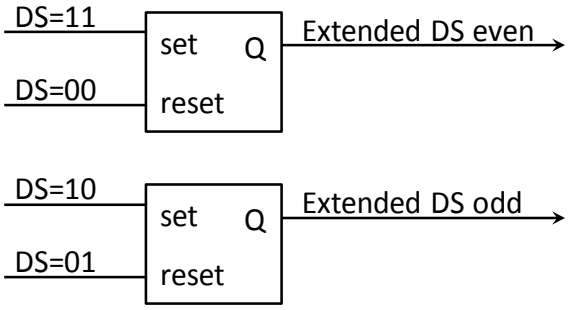


Figure 7: Extended even/odd state

Race condition free decoder logic is shown in the figure 8. Since the extended even/odd states are held for the duration of two bits, the logic of this stage works at higher bit rate compare to the simple decoder logic. However, merging even states and odd states to obtain the "character", only one bit duration is allowed. Consequently, maximum bit rate is the same as the simple decoder logic but not less.

4 SUMMARY

Because of race condition, the simple DS decoder logic is not portable. In order to avoid race condition, input states are divided into two groups using DS parity. Even parity state and odd parity state appear alternatively. They can be handled without race condition. This race condition free decoder logic works at maximum speed of implemented FPGA.

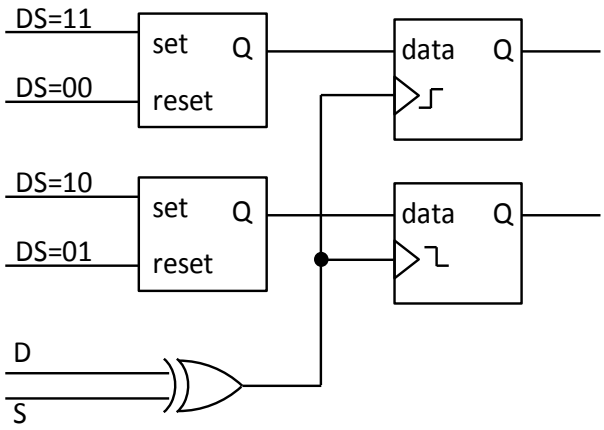


Figure 8: Race condition free decoder logic

Poster Presentations

VALIDATION AND TESTING OF AN IP CODEC FOR HIGH BANDWIDTH SPACEWIRE LINK[1]

Session: SpaceWire Test and Verification (Poster)

Short Paper

R. Castillo, J. Martín, J. Almena, M. Prieto, D. Guzmán, S. Sánchez

Space Research Group. Dpto. Automática. Universidad de Alcalá. Ctra. Madrid Barcelona Km 33,600. 28871, Alcalá de Henares (Madrid), Spain

P. Aguilar-Jiménez

INTA. Carretera de Ajalvir, km. 4. 28850, Torrejón de Ardoz (Madrid), Spain

E-mail: rcastillo@srg.aut.uah.es, jamartin@srg.aut.uah.es, jalmena@srg.aut.uah.es, mpm@srg.aut.uah.es, dguzman@srg.aut.uah.es, chan@srg.aut.uah.es, aguilarjp@inta.es

ABSTRACT

In this paper we present the design, development and testing of a SpaceWire codec that is fully compliant with ESA standard ECSS-E-ST-50-12C. This codec is part of the high bandwidth communication infrastructure employed in the Spanish INTA Microsat satellites programme. Four FPGAs families have been used to validate our design. Three of these implementations have been tested against commercial solutions using a suite of utilities developed within our group. With these utilities, the user is able to configure the hardware, transfer data and check the status of the SpaceWire links. The results of all tests are presented, including real performance results and compatibility test results.

1 INTRODUCTION

The Space Research Group of the Universidad de Alcalá has developed a SpaceWire IP Codec that allows a high bandwidth data exchange through a SpaceWire link. The initial motivation for the development of this IP Codec was to satisfy the data communications requirements for the Energetic Particle Detector (EPD) onboard Solar Orbiter [1]. It is also planned to use it within the MicroSAT satellites programme, which belongs to the Spanish Instituto Nacional de Técnica Aeroespacial (INTA) [2].

In the process, the Codec has been developed for four FPGA families, namely Spartan3E and Virtex4 from Xilinx and ProASIC3E and Axcelerator from Actel. Several tests have been carried out for most of these families, including compatibility with existing commercial solutions from Star Dundee and Gaisler Research, data transfer rate and data integrity tests.

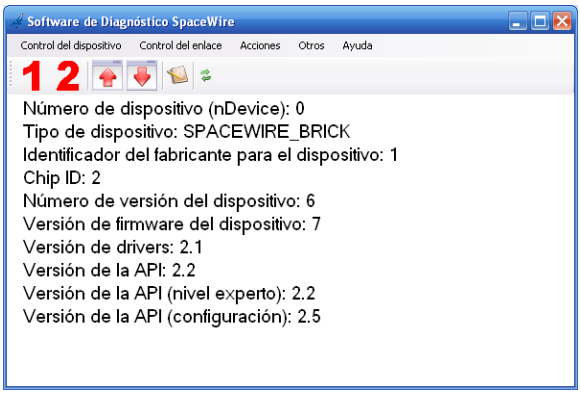
[1] This work has been supported by the MICINN grant AYA2009-13478-C02-02.

In this paper, the results of all tests made are available, as well as occupation information for the SpaceWire Codec developed by our group compared to similar products developed by others.

2 TESTING ENVIRONMENT

To carry out the aforementioned tests, our SpaceWire Codec was tested against three commercial solutions: Star Dundee's SpaceWire PCI-2, Star Dundee's USB Brick and Gaisler Research's RTC Development Suite.

For controlling both of Star Dundee's solutions, a Microsoft Windows diagnostic application was developed by our group. It allows the user to configure the hardware, send and receive data, perform loopback tests, as well as get information about the status of the SpaceWire links and data transfers. The main screen of the application is shown in the figure on the right side of these lines.



For performing the tests against the Gaisler Research's RTC, several applications have been developed for the RTEMS Operating System [3]. These applications allow sending and receiving data over a SpaceWire link, measuring the time used to carry out the transmission and comparing data sent and received so that they are equal, allowing for a data integrity test. A screen capture of the application is shown in figure 1.

```
grlib> load t
section: .text at 0x40000000, size 128064 bytes
section: .data at 0x4001f440, size 2784 bytes
total size: 130848 bytes (87.8 kbit/s)
read 743 symbols
entry point: 0x40000000
grlib> run
******** Starting SPW TX TEST ********
LINK_STATE[SPW0]: 5 - RUN
SpaceWire Transmission FINISHED: 16387,2 Mb in 133.0000 s [123.2120 Mbs]

Program exited normally.
grlib>
```

Figure 1: Application for controlling Gaisler Research's RTC.

3 TEST RESULTS

For testing and validating our IP Codec implementations, three kind of tests were carried out:

a) PC/RTC to FPGA and FPGA to PC/RTC data transmission: This test allows to determine the maximum data rate achievable. The FPGA is programmed with our SpaceWire CODEC, plus a component that generates data and sends it through the SpaceWire Codec, and a data receiver component which reads the data that arrives from the SpaceWire link through the Codec.

In this test, a stream of data is sent from the PC/RTC or FPGA to the other end of the link, and the time it takes to transfer the data is measured.

b) Data Integrity test: For this test, the FPGA is programmed with our SpaceWire Codec, plus a FIFO Buffer and an interface component. This interface component has two tasks: first, it reads data that arrives from the SpaceWire link through the Codec and stores it in the FIFO Buffer. When there is data available in the buffer, this component reads it and sends it through the Codec.

   In this test, a PC sends several fixed-sized packets through the SpaceWire link, filled with random data. The IP Codec resends the data to the PC upon reception. In the PC, the data received is compared with the data sent to check if there are errors.

c) Loopback test: This test is similar to the first one, but the Codec outputs are connected with its inputs. This allows to determine the maximum operating frequency of the Codec. Also, when sending data through the link, it allows to determine the maximum data rate for simultaneous transmit and receive operations.

General testing results are presented on figure 2.

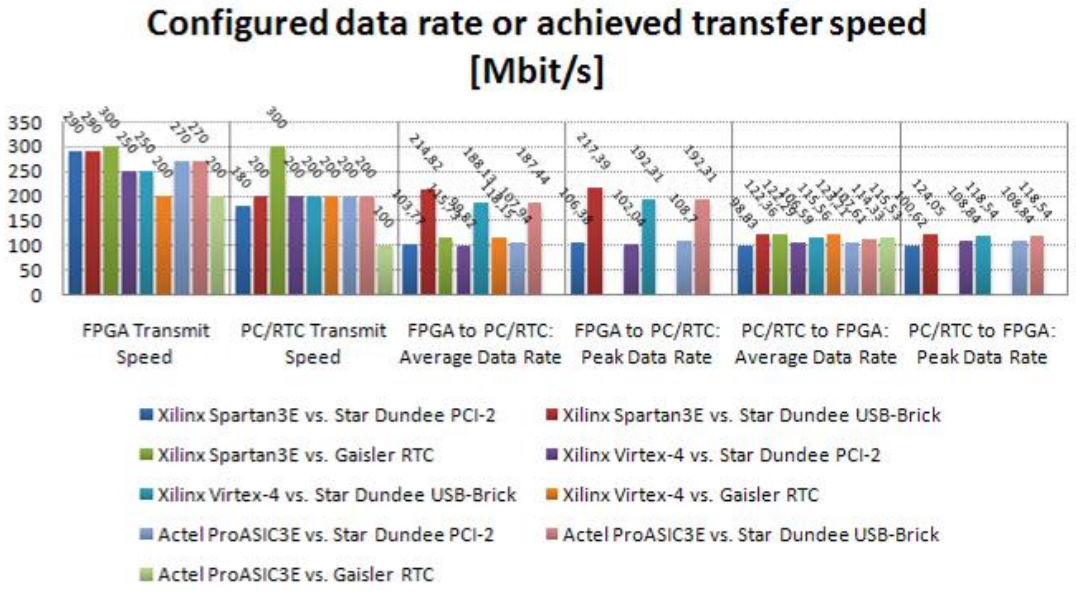


Figure 2: SpaceWire IP Codec test results

Other tests were also carried out for the Xilinx Virtex4, Xilinx Spartan3E and Actel ProASIC3E implementations:

- Data Integrity: Test carried out successfully on all implementations, transferring and comparing 25.000 packets of 21.844 bytes each.

- Loopback Test: Link established successfully on all implementations, with a transmitter speed of 290 Mbit/s for Xilinx Spartan3E, 250 Mbit/s for Xilinx Virtex4 and 270 Mbit/s for Actel ProASIC3E. Data throughput was measured

to be 218,45 Mbit/s for Spartan3E, 190,512 Mbit/s for Virtex-4 and 204,8 Mbit/s for ProASIC3E.

- Post Place&Route Simulation: Two instances of the FPGA design explained in the data transmission test were tested against each other. A SpaceWire link was established successfully on all implementations, at 300 Mbit/s for Spartan3E, 400Mbit/s for Virtex4 and 270 Mbit/s for the ProASIC3E implementation. Exchange of data was carried out normally in all cases.

3.1 ACTEL AXCELERATOR

This implementation has only been tested on post-layout simulation. Two instances of the FPGA design explained in the data transmission test were tested against each other at 200 Mbit/s. A link was established successfully and a transfer of a 20480 byte packet took 1075,761 µs, which results in a data rate of 152,301 Mbit/s.

4 OCCUPATION DATA

In figure 3, the reader can find the occupation data for the four implementations of the SpaceWire Codec, as well as data for alternative solutions from AeroFlex Gaisler and ESA. Information from Star Dundee is also shown below. Please keep in consideration that the data from these alternative solutions could also include additional functionality besides the SpaceWire Codec.

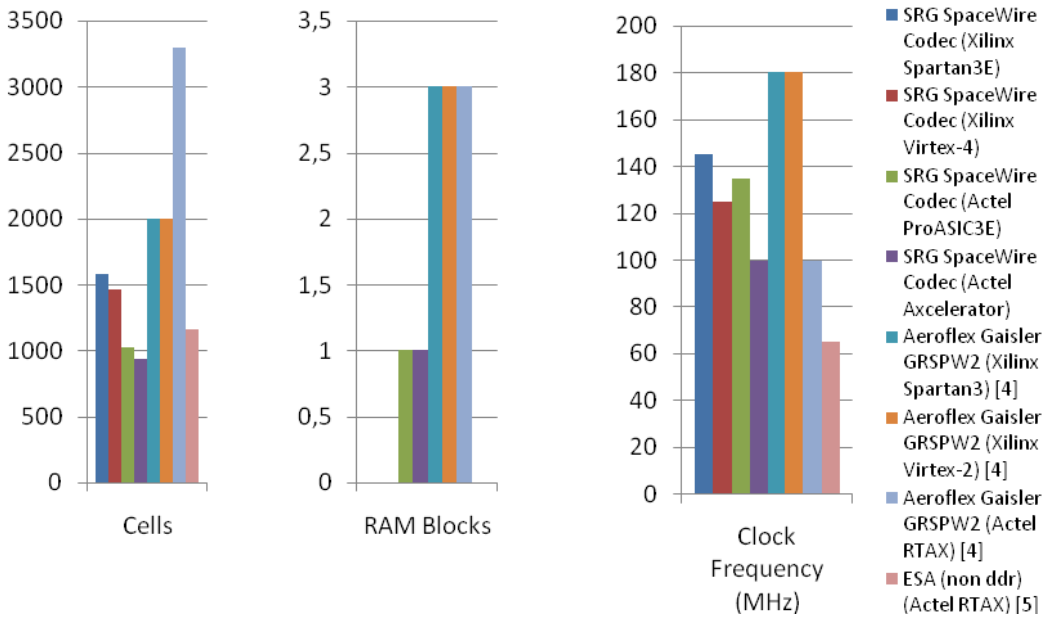


Figure 3: SpaceWire IP Codec occupation data

- Star Dundee [6]: Approximately 7% of an Actel RTAX1000 or 4% of a Xilinx Spartan3E 1600

5 CONCLUSIONS AND FUTURE WORK

A SpaceWire IP Codec has been developed for four of the most used FPGA families, which allows for a high bandwidth point to point data exchange. Compatibility tests of our SpaceWire Codec with current commercial alternatives have been carried out

successfully, reaching data throughputs of up to 217,39 Mbit/s on a Xilinx Spartan3E, 192,31 Mbit/s on a Xilinx Virtex-4 and 192,31 Mbit/s using an Actel ProAsic3E. On simulation, an Actel Axcelerator implementation reached 152,301 Mbit/s.

Work is in progress to enhance the IP Codec's functionality, adding support for Packet Routing, RMAP [7] and CCSDS [8], as well as increasing the maximum operating speed, especially for the Xilinx Virtex implementation where the maximum operating frequency in simulation was greater than the maximum frequency achievable for a normal operation in the FPGA implementation.

6 REFERENCES

1. ESA Science & Technology: Instruments

   http://sci.esa.int/science-e/www/object/index.cfm?fobjectid=40396

2. Instituto Nacional de Técnica Aeroespacial

   http://www.inta.es

3. RTEMS Operating System

   http://www.rtems.com

4. Aeroflex Gaisler. GRSPW2 Spacewire Codec

   http://www.gaisler.com/cms/index.php?option=com_content&task=view&id=276&Itemid=141

   Retrieved on March 31st, 2010

5. European Space Agency. Spacewire-b

   http://www.esa.int/TEC/Microelectronics/SEMLOU8L6VE_0.html

   Retrieved on March 31st, 2010

6. Star Dundee. SpaceWire IP from STAR-Dundee

   http://www.star-dundee.com/content/products/003~~Other%20Products/datasheets/SpaceWire%20IP.pdf

   Retrieved on March 31st, 2010

7. SpaceWire Remote Memory Access Protocol

   http://spacewire.esa.int/content/TechPapers/documents/SpaceWire%20RMAP%20DASIA%202005.pdf

8. ECSS-E-ST-50-53C SpaceWire - CCSDS packet transfer protocol

   http://www.ecss.nl/forums/ecss/dispatch.cgi/standards/docProfile/100771/d20100209121732/No/t100771.htm

FORMAL VERIFICATION FOR SPACEWIRE LINK INTERFACE USING MODEL CHECKING

Session: SpaceWire Test and Verification (Poster)

Long Paper

Zhiquan Dai

College of Information Engineering, Capital Normal University, Beijing, China

Limin Tao

Beijing Engineering Research Center of High Reliable Embedded System

Beijing, China, 100048

Liya Liu

Dept. of Electrical and Computer Engineering, Concordia University

1455 de Maisonneuve W., Montreal, Quebec, H3H 1M8, Canada

Yong Guan, Weigong Zhang, Yuanyuan Shang, ShengZhen Jin

College of Information Engineering, Capital Normal University, Beijing, China

E-mail: guanyong@mail.cnu.edu.cn, woyun_23@163.com, liy_liu@ece.concordia.ca, zwg771@yahoo.com, syy@bao.ac.cn

ABSTRACT

The design of the SpaceWire based satellite onboard system circuits was a part of the job in the development of Space Solar Telescope (SST) project, which has been completed by National Astronomical Observatories, Chinese Academic of Sciences. In order to prove the circuit was faithfully implements the SpaceWire protocol's specification, formal verification techniques were applied during the process of development of the circuits and automated model checking approach was employed. The implementation designed as VHDL models on the FPGA for SpaceWire link interface circuit under investigation has an extension state (Error Analysis) in the state diagram providing link initialization, normal operation and error recovery services between transmitter and receiver on exchange level. Some properties were checked successfully on the original model by using Cadence SMV tool and some properties were verified to false. The results of the verification showed we have to update the design according to the counterexamples to guarantee the circuit design implemented on FPGA is reliable and can be integrated in the SST project.

1 INTRODUCTION

The correctness of design is one of the key problems to large-scale complex digital system design, namely, design verification. Unfortunately, the complexity of the verification exponentially increases with the increasing scale of chip. Particularly, as complex the state of the art is, the cost of trial-produce is quite expensive. Safety is the first place for many very important systems, for instance, the railway signal, nuclear power station, aerospace, national security and large communication system [1]. Any mistake of design possibly causes huge economic losses or catastrophic consequences as personnel casualties. The design of the SpaceWire based satellite onboard system circuits was a part of the job in the development of Space Solar Telescope (SST) project, which has been completed by National Astronomical Observatories, Chinese Academic of Sciences. In order to prove the circuit designed for the highly reliable communication based on SpaceWire protocol was faithfully implements the SpaceWire protocol's specification, this study aimed to verify the SpaceWire link interface, which was one of the important elements of the SpaceWire. Formal verification techniques were applied during the process of development of the circuits and automated model checking approach was employed.

Techniques for automatic formal verification of finite state transition systems have developed in the last 30 years to the point where major chip design companies are beginning to integrate them in their normal quality assurance process. The most widely use of these methods is called Model Checking [9]. In model checking, the design to be verified is modelled as a finite state machine, and the specification is formalized by writing temporal logic properties. The reachable states of the design are then traversed in order to verify the properties. In case that the property fails, a counter example is generated in the form of a sequence of states [7]. In general, properties are classified to "safety" and "liveness" properties. The former declares what should not happen or what should always happen; the latter declares what should eventually happen. Specification is a process to briefly express the design system and its properties with formal language. Formal specification description language has strict syntax and semantics, which are used to express the functional behavior of the system, such as timing characteristics or internal structure.

The main fault of traditional testing and simulation verification is that they are incomplete. In another words, they can only prove that the design has error but can not guarantee the design has no error. So, they are often suitable to find the vast or obvious errors in the initial verification, but not to find complex and subtle errors [1]. The main advantage of the formal verification is completeness. Through model checking, a method of the formal verification, we can find complex or subtle design mistakes which other methods cannot find. So, model checking is an effective way of the computer system design verification.

2 APPROACH

2.1 VERIFICATION FLOW

The overall flow of our approach is depicted in Figure1. The verification flow is also applicable for other classes of circuit verified by model checking methods.

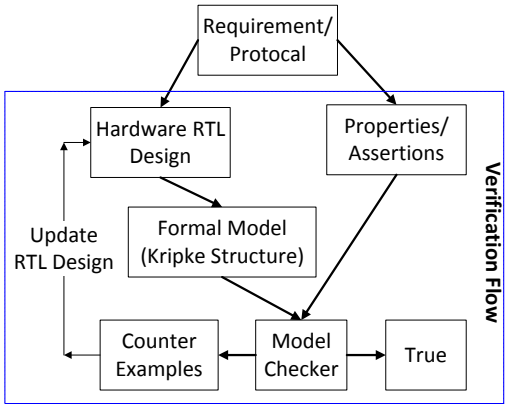


Figure1. Formal Verification Flow

The inputs to the process are the RTL description of the circuit, a formal specification (possibly comprising many properties/assertions). We elaborate on the latter point, for Properties/Assertions of SpaceWire control, in Section 3.2. The formal SpecaWire control model is automatically compiled into a finite state machine [11].

The RTL is translated into a formal model of the circuit, either manually or using automated tools. For the work in this paper, this is a description in the input language of a model checker. (The model checker Cadence SMV [2] includes an automated translator from Verilog to its input format.) However, in general it would depend on the formal verification (FV) tool that is used. If the verification result is false, then update the RTL design according to the counterexamples generated by the model checker automatically.

2.2 FORMAL MODEL

The hardware engineer's design is usually some sort of a finite automaton. Independent of the concrete design language, this finite automaton can be represented by a kripke structure, which is the standard representation of models in the model checking literature [9]. The kripke structure is a quintuple K=(S, S_0, R, AP, L), where S is the finite state set of all the Boolean state variables $\{s_1, s_2 \cdots s_n\}$, $S_0 \subseteq S$, denoting the set of initial states in which the circuit can begin operation, R is the transition relation of the system defining how the system evolves over time, AP is the set of all the atomic proposition and its negative proposition, and L is the marking function which maps the state $s \in S$ into the true atomic proposition set of S [1]. We can also regard K as a marked directed graph with a root, S is the vertex set of the graph, R is the edge set of the graph, L is marking function of the vertex, and the root is s_0.

Given an RTL-level circuit designed with hardware description language, for example Verilog, a formal model will be created automatically by the X-HDL tool as mentioned above [10]. Timing-related details in the RTL are modelled using non-determinism, so that the resulting formal model exhibits a superset of the actual system behaviours. Any verification performed on the formal model will then be faithful with the specification.

3 SpaceWire Control Module

SpaceWire[8] is a network for space applications composed of nodes and routers interconnected through bi-directional high speed data links. According to the SpaceWire website hosted by the ESA, it has been used in missions of the ESA as well as space agencies NASA and JAXA. The SpaceWire standard [8] describes 6 protocol levels: physical, signal, character, exchange, packet, and network. In this paper, we concern with the exchange level that defines the protocol for link initialization, flow control, and link error detection and recovery (similar to the more widely known Transmission Control Protocol, TCP). Our main case study is the SpaceWire control module of a node in the SpaceWire network, which is implemented by our group in VHDL description language. Unfortunately, as VHDL is not the required input language of any model checking tool available to us at this moment, the design is translated from VHDL into Verilog by X-HDL [10]. With aid of this tool, code in Verilog was automatically translated into the input code with the acceptable language for the Cadence SMV model checker. In the mean time, English language specifications from the standard document [8] were translated into formal specifications in linear temporal logic and inserted into the SMV file as assertions to be checked [3].

3.1 Model

A SpaceWire end node comprises three modules: a transmitter (TX), a receiver (RX), and a state machine that sends control signals to them (FSM). We abstracted the control module code from our whole design. Generating a SMV model from Verilog involved straightforward transition for the most part, retaining the control structure, and only abstracting away some data and timing in the Cadence SMV checking tool [2]. FSM module indicating how state was abstracted to be the SMV model is briefly described the following part [4]. Further details may be found in the standard document [8].

The FSM controls the overall operation of the end node. Its operation is shown in Figure2. The sequence of state ErrorReset, ErrorWait, and Ready provide a mechanism of initializing the SpaceWire node, either coming from a whole system reset or triggered by an error. During this sequence of operation, RX is enabled to receive, but TX is prohibited from sending. In the Started state, TX can send NULL signals to the other end, to establish a connection. Next, the FSM enters the Connecting state where TX is enabled to send flow control tokens (FCTs). When RX

receives FCTs, it indicates that the other end has space in its receive buffer for data. The Run state is the state for normal operation where packets flow freely in both directions across the link. The node remains in the Run state until an error occurs or until the link is disabled [8]. An ErrAnalysis_DataSave state was added in order to improve the error analysis and process ability. When an error occurs or the link is disabled in the run state, FSM enter into ErrAnalysis_DataSave state. In the same time, FSM save and analyze the error and the data. If the data has been saved and the error has been read, then the FSM enter into ErrorReset state, or still in the ErrAnalysis_DataSave state.

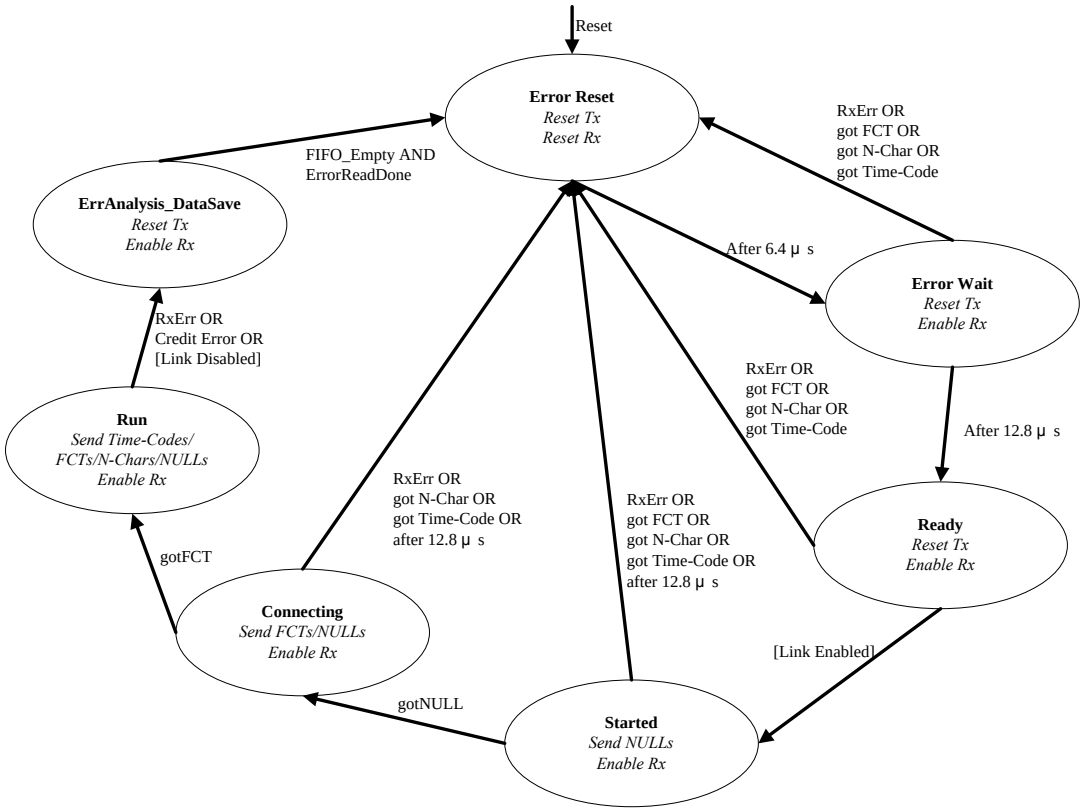


Figure2. SpaceWire Control Module State Graph

The end nodes communicate over a channel that was modelled in SMV to be capable of dropping or creating parity errors in both control and data packets. (Appropriate "fairness" constraints [6] were imposed on the channel to ensure that a packet would eventually get to its destination, even if it is dropped several times.)

3.2 FORMAL SPECIFICATIONS

25 SMV assertions were added in linear temporal logic corresponding to the specification written in nature language in the protocol [8]. Temporal logic formulas presented as formulas in ordinary Boolean logic, except that true value of a formula in temporal logic is a function of time. Some new operators are added to the traditional Boolean operators "and", "or", "not" and "implies", in order to specify relationships in time. The new operators are termed as tense operator consists of *G* (global), *F* (future), *X* (next) and *U* (until). *G p* means that *p* will keep true all the times in the future and is

read as "eventually *p*". The formula *F p* express *p* must hold true at some time in the future and is read as "eventually *p*". In addition, we have the "until" operator and the "next time" operator. The formula *X p* means that *p* will be true at the next time and is read as "next *p*". The formula *p U q* means that *q* is eventually true, and until then, *p* must always be true and is read as "*p* until *q*" [3].

No	Reference in[9]	Assertion
1	Sec. 8.5.2.2(c)	assert G(!Reset & After64 &SpacewireControllerCurrentState = 0 -> X(SpacewireControllerCurrentState = 1)); --When the reset signal is de-asserted the ErrorReset state shall be left unconditionally after a delay of 6,4 μs (nominal) and the state machine shall move to the ErrorWait state.
2	Sec. 8.5.2.3(b)	assert G(SpacewireControllerCurrentState = 1 -> X(!RX_Reset & TX_Reset)); -- In the ErrorWait state the receiver shall be enabled and the transmitter shall be reset.
3	Sec. 8.5.2.4(a)	assert G(X(SpacewireControllerCurrentState = 2) ->(SpacewireControllerCurrentState = 1 | SpacewireControllerCurrentState = 2)); --The Ready state shall be entered only from the ErrorWait state.
4	Sec. 8.5.2.5(g)	assert G(SpacewireControllerCurrentState = 3 & (DisconnectionError | FirstNULLreceived_internal & (RX_Error | RX_GotSomethingWrong)) ->X(SpacewireControllerCurrentState = 0)); --If, while in the Started state, a disconnection error is detected, or if after the gotNULL condition is set, a parity error or escape error occurs, or any character other than a NULL is received, then the state machine shall move to the ErrorReset state.

Table1. Selected Formal Specification

Table1 lists the representative assertions. Specifications are classified into four categories, and each category is represented in the table. The first set of specifications is on the FSM operation, indicating how and when the system can move between FSM states, as shown in Figure2. The second set is related on the cases whether the transmitter and receiver are enabled or not. The third presents the situations that current state is transferred from itself or the previous state. The fourth is based on the interaction between FSM, TX, and RX, exemplified by row 4 in the table that deals with error handling. Our formal specification is as comprehensive as the corresponding English language specifications in the standard documents.

3.3 RESULTS

An SMV model with 333 lines code (including assertions) was generated based on a design with 511 lines code Verilog language. According to the SpaceWire protocol,

the formal specifications are created, the state transition properties indicating that the current state is from itself or the previous state is verified to be true. The error handling properties are verified to be false. For instance, the assertion, indicating that if any error occurs, the ErrorWait state will move into the ErrorReset state in the SpaceWire standard document section 8.5.2.3 (e), is verified to be false. The counterexample of it shows that the error occurs, but the FSM do not move into the ErrorReset state. It might be the result of the case that our assertion is not stated as the specification of SpaceWire protocol or the RTL designed by our group really has some error. We will continue to consummate our formal verification and update our RTL design according to the verification results.

4 CONCLUSIONS AND FUTURE WORK

The translation from VHDL to SMV is automatically. Some properties of SpaceWire control module were verified successfully, and the remaining properties will be verified in the future. Although the system of assertions is quite comfortable, we have to study the informal descriptions of parts of the design and often formulate our own assertions to verify the design, because hardware designers are often not aware of all presumptions they use to believe that their source codes are correct. Although the model checking verification is complete, it is easy to generate the state space combination explosion problem. Users of model checking tools typically consider it a compliment to the traditional methods of testing and simulation, and not as an alternative.

ACKNOWLEDGMENTS

We appreciate Kenneth L. McMillan for help with Cadence SMV. And we would like to acknowledge our group for providing the SpaceWire RTL design and Maning Wan for helping us with understanding the SpaceWire protocol. This research was supported in part by the Space Solar Telescope (SST) project.

REFERENCES

1. Han Jungang, Du Huiming, “Digital hardware formal verification”, Peking University press, 2001.

2. Cadence SMV model checker, http://www.kenmcmil.com/smv.html.

3. K.L.McMillan, “Getting started with SMV”, SMV Reference Manual, Cadence Berkeley Labs, Berkeley, 1999.

4. K.L.McMillan, “The SMV language”, SMV Reference Manual, Cadence Berkeley Labs, Berkeley, 1999.

5. K.L.McMillan, “The Model Checking System” SMV Reference Manual, Cadence Berkeley Labs, Berkeley, 2002.

6. K.L.McMillan, “Symbolic Model Checking”, Kluwer Academic Publishers, 1992.

7. E. M. Clarke, O. Grumberg, D. A. Peled, “Model Checking”, MIT Press, 2000.

8. European Cooperation for Space Standardization, “Space engineering. SpaceWire - links, nodes, routers, and networks”(ECSS-E-50-12A), http://www.ecss.nl/forums/ecss/dispatch.cgi/standards/showFile/100302/d20090722143301/No/ECSS-E-50-12A(24January2003).pdf, 2003.

9. Sanjit A. Seshia, Wenchao Li, Subhasish Mitra. “Verification-guided soft error resilience”, Design, Automation, and Test in Europe, 2007, Pages: 1442 – 1447.

10. Tomáš Kratochvíla, Vojtěch Řehák, Pavel Šimeček, “Verification of COMBO6 VHDL Design”, CESNET, 2003.

11. Tahar, “Temporal Logics and Model Checking”, http://users.encs.concordia.ca

    /~tahar/coen7501/notes/3-mc-02.05.pdf.

IMPLEMENTATION OF A DYNAMICALLY RECONFIGURABLE PROCESSING MODULE FOR SPACEWIRE NETWORKS

Session: SpaceWire Onboard Equipment and Software (Poster)

Short Paper

Florian Dittmann, Markus Linke

TWT GmbH Science & Innovation, Bernhäuser Str. 40, 73230 Neuhausen, Germany

Jens Hagemeyer, Markus Koester, Julien Lallet, Christopher Pohl, Mario Porrmann

Heinz Nixdorf Institute, Fürstenallee 11, 33102 Paderborn, Germany

Julian Harris

Swiss Space Technologies, Route de Chavalet 2, Champéry, Switzerland

Jørgen Ilstad

European Space Agency, Postbus 299, NL-2200 AG Noordwijk, The Netherlands

E-mail: research@twt-gmbh.de, porrmann@hni.uni-paderborn.de, julian@swissspacetech.com, jorgen.ilstad@esa.int

ABSTRACT

The ESA-project "FPGA based generic module and dynamic reconfigurator" targets the development of a hardware architecture, called DRPM (for Dynamically Reconfigurable Processing Module). The goal of the DRPM is to develop a system that allows for the adaptation of hardware components in flight at run-time. This is enabled by the implementation of an SRAM-FPGA-based partially reconfigurable core, which is embedded into a system hosting a reconfiguration controller and a system controller providing suitable interfaces for space applications. Maximum flexibility is realized by implementing SpaceWire interfaces that enable the DRPM integration into a SpaceWire network. Moreover, the SpaceWire RMAP protocol is used for remote access to registers and memory banks of the DRPM.

1 INTRODUCTION

One of the main advantages of the SpaceWire protocol is the reusability of already developed SpaceWire-based memory units and payload processing components. The reusability does not only reduce the development costs, but also improves the reliability of the systems. Therefore, there is a need for increasing the reusability of already developed payload processing components for future missions. With respect to hardware devices, SRAM-based FPGAs offer reconfigurability, which allows for the development of generic payload processing modules. The use of generic modules in a space environment has numerous advantages. In terms of system-specific hardware, a reconfigurable generic module can ensure maximum reuse of development across different systems. Since high development costs are usually incurred during custom FPGA development, a more modular approach can be used with only custom modules being added as peripheral systems within the FPGA.

Current and upcoming application specific standard products (ASSPs) offer limited processing performance, which require multiple device instances in many cases to support the high data rates provided by current sensors. In contrast to ASSPs, FPGAs are advantageous for common high data rate applications like image processing, compression, and generic signal processing. Compared to general purpose processors, FPGAs often offer a better performance and more adaptability, particularly for applications where a parallelized algorithm can be implemented on the FPGA.

Reconfigurable hardware allows payload processing to be changed or adapted during a mission. Therefore, systems can be freely adapted to every possible scenario, even if it was not foreseen at design time. Moreover, if the devices can be reconfigured dynamically, this adaptation can also be done in an efficient manner during run-time, where parts of the device can remain operative, whilst others are changed. This allows for implementing time-sharing of the reconfigurable resources between different applications, thus increasing the area efficiency.

The following sections show the DRPM in detail, emphasizing its SpaceWire functionality, which is core to the DRPM concept in terms of payload system integration. The SpaceWire capabilities of the DRPM are also complemented by additional interfaces for directly accessing high speed instruments.

2 DYNAMICALLY RECONFIGURABLE PROCESSING MODULE

The focus of the project is on the development of the reconfigurable core including its control mechanisms. The DRPM demonstrator aims at showing the capabilities of the reconfigurable core by executing different payload processing applications on the same hardware. Concerning the harsh space requirements, mitigation of radiation effects and recovery in case of failure is emphasized.

The DRPM features several SpaceWire interfaces for avionics and instrument source data allowing for the DRPM to be integrated into a SpaceWire network as general purpose processing node. The DRPM supports the SpaceWire remote memory access protocol (RMAP) [1] in order to enable access to the registers and memory banks of the DRPM from other nodes in the SpaceWire network. The SpaceWire RMAP functionality allows for access to the working memory of the payload processing components as well as access to the program and configuration memory. This is useful for debugging purposes or remote uploading of new software applications or FPGA configuration files.

Besides its SpaceWire connectivity, the DRPM features additional dedicated instrument interfaces, ranging from high rate instruments interfaced via WizardLink to lower rate instruments interfaced via CAN bus, discrete I/O and legacy serial links such as e.g. RS422. Instrument control may be performed according to an instrument’s specific needs, which may include control via SpaceWire, CAN bus or customized discrete interfaces. As an alternative to SpaceWire the avionics can be connected via MIL-STD-1553B. The DRPM contains a partially reconfigurable core that allows for customized interfaces to be tailored depending on specific needs. The main feature of the partially reconfigurable core, however, is to allow for high performance data processing algorithms to be implemented to cover a wide range of applications. In addition, it shall support in-flight reconfiguration during a mission where required, whilst being fault-tolerant to space environment effects typically

caused by high energy particles. A demonstrator platform of the DRPM is being developed, which is used for system exploration. It is based on the rapid prototyping system RAPTOR [2] and features all required hardware components and interfaces. Fault injection mechanisms, which emulate effects such as single event upsets, are used to verify the fault-tolerance features of DRPM.

3 PARTIAL FPGA RECONFIGURATION

The DRPM consists of partially reconfigurable FPGAs (PR FPGAs), which are used for the payload processing. When utilizing partial reconfiguration suitable design methods are required for partitioning the FPGA resources. In the DRPM a tiled partially reconfigurable region as described in [3] is used, where the FPGA area is divided into a static region and a partially reconfigurable region (PR region). The static region contains system components that are constantly active, while the PR region is reserved for dynamic system components, which are referred to as partial reconfiguration modules (PR modules). PR modules can be loaded and unloaded at run-time according to the needs of the application. The partial reconfiguration is performed by a dedicated reconfiguration controller, which transfers the configuration files from the PR module storage to the configuration interface of the PR FPGA. In order to mitigate single-event upsets (SEU), hardening techniques such as triple modular redundancy and configuration memory scrubbing are applied.

4 SYSTEM EXPLORATION

In the following, two example systems are introduced, which can be realized by the DRPM demonstrator platform. Both systems feature avionics and source data interfaces as specified in Section 2. The building blocks are the SpaceWire RTC AT7913E, one or more partially reconfigurable FPGAs (PR FPGAs), and a communication FPGA (COM FPGA), which is used to interconnect the SpaceWire RTC and the PR FPGAs.

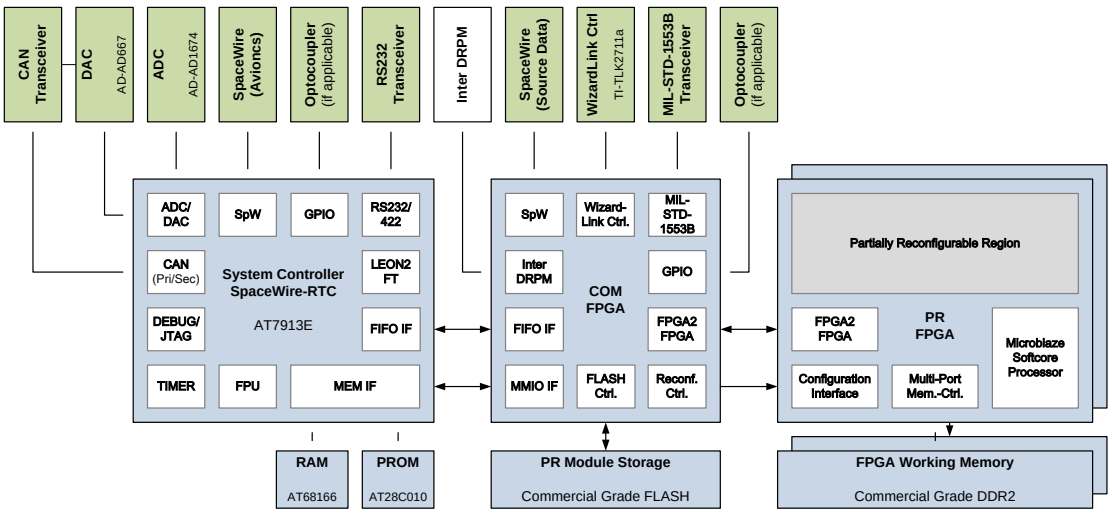


Figure 1 DRPM with external device interface control via the COM FPGA.

In the first system, shown in Figure 1, the interface control is implemented in the COM FPGA. It includes source data instrument interface controllers, such as 4 SpaceWire RMAP IP Cores [4], a Wizard-Link controller, general purpose IO, and a

MIL-STD-1553B controller. The system controller is connected via a FIFO and a memory mapped IO interface. The COM FPGA implements a Flash controller to gain access to the PR module storage. The reconfiguration controller is used to apply full and partial reconfiguration of the PR FPGA. The inter DRPM interface allows a connection to additional DRPMs. The second system, which is shown in Figure 2, implements the interface control on the PR FPGA. The only components, which are implemented on the COM FPGA are the FIFO interface to the system controller, the reconfiguration controller, and the corresponding configuration memory controller. In comparison to the external device interface control, the self-hosting interface control allows for a closer coupling between source data interfaces and the PR modules used for payload processing. The self-hosting interface control requires fewer resources on the COM FPGA, but more resources on the PR FPGA.

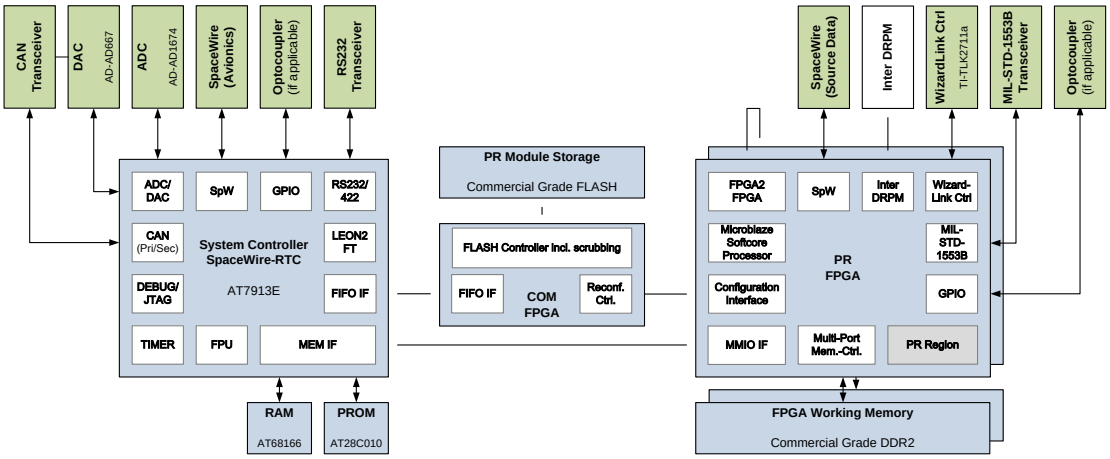


Figure 2 DRPM with self-hosted interface control on the PR FPGA.

5 CONCLUSION

In the current state of the project, the hardware and software components of the DRPM demonstrator are under development. One of the main features is the utilization of dynamically reconfigurable FPGAs, which are used for high performance payload processing. The SpaceWire RMAP protocol is used to remotely control the DRPM in a SpaceWire network. The flexibility of the DRPM demonstrator helps identification of different system architectures for space applications. Thus, various systems can be prototyped and analysed before defining the final architecture, which is considered for flight use.

6 REFERENCES

[1] “SpaceWire - Remote Memory Access Protocol”, ECSS-E-ST-50-51C, 2010.

[2] “A Prototyping Platform for Dynamically Reconfigurable System on Chip Designs”, H. Kalte, M. Porrmann and U. Rückert. In: Proceedings of the IEEE Workshop Heterogeneous reconfigurable Systems on Chip (SoC), 2002.

[3] “Design of Homogeneous Communication Infrastructures for Partially Reconfigurable FPGAs”, Jens Hagemeyer, Boris Kettelhoit, Markus Koester and Mario Porrmann, In: Proceedings of the International Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA '07), p. 238-247, 25-28 June 2007.

[4] “SpaceWire RMAP IP core”, S. Parkes, C. McClements, M. Dunstan and W. Gasti, In: Proceedings of the Second International SpaceWire Conference, 4-6 November 2008.

DEVELOPMENT OF SPACEWIRE BASED DATA ACQUISITION SYSTEM FOR THE X-RAY CCD CAMERA ON BOARD ASTRO-H

Session: SpaceWire Missions and Applications (Poster)

Short Paper

Takahisa Fujinaga, Masanobu Ozaki, Tadayasu Dotani, Hirokazu Odaka, Tadayuki Takahashi, Motohide Kokubun, Takeshi Takashima,

Institute of Space and Astronautical Science/JAXA, 3-1-1 Yoshinodai, Chuo-ku, Sagamihara, Kanagawa 252-5210 JAPAN

Naohisa Anabuki, Hiroshi Nakajima, Kiyoshi Hayashida,
Hiroshi Tsunemi, Masaharu Nomachi,
Osaka University, 1-1 Machikaneyama, Toyonaka, Osaka 560-0043 JAPAN

Shoichi Aoyama, Koji Mori,

University of Miyazaki, 1-1 Gakuen Kibanadai Nishi, Miyazaki 889-2192, JAPAN

Takayuki Yuasa

The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-0033, JAPAN

E-mail (TF): fujinaga@astro.isas.jaxa.jp

ABSTRACT

We present the development status of the ground-support digital data acquisition system for the X-ray CCD camera (SXI: Soft X-ray Imager) onboard the ASTRO-H satellite. ASTRO-H employs SpaceWire (SpW) as its information network overall, and the CCD data are also acquired through it. We have developed the data acquisition system for the SXI BBM (breadboard model). In the system, the digitized data are are first stored in the SDRAM of the DE I/F, digital circuit board with SpW interface. Then, they are transferred to a SpaceCube1 (SpC) and finally received by a POSIX-OS computer. The DE I/F board and the SpC are connected by a SpW, and the SpC and the POSIX machine is connected by an Ethernet. We used the SpC as a protocol converter between SpW and TCP/IP, and run the DAQ software on the POSIX system. The DAQ system was stable more than 24 hours. The mean transfer speed was, however, 4.4 Mbps, which is slower than the system requirement 8.8 Mbps. We succeeded in obtaining and storing the CCD images continuously by this system with the analogue part of the BBM.

1 INTRODUCTION

ASTRO-H is an X-ray observatory scheduled to be launched on 2014. It carries four kinds of science detectors covering the 0.1-600 keV energy band, and is expected to reveal various high energy phenomena. The information network of ASTRO-H is based on SpaceWire (SpW) (see [1]).

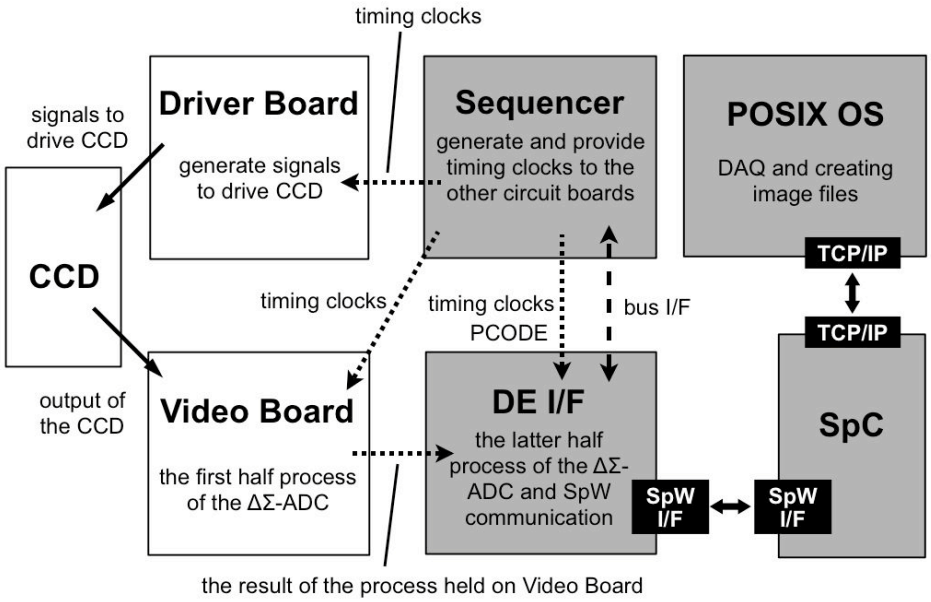


Figure 1: A block diagram of the SXI BBM, which consists of circuit boards of Driver Board, Video Board, Sequencer and DE I/F. The solid, the dotted, and the dashed arrows show analogue signals, LVDS, and LVTTL respectively. The greyed boxes are the digital components discussed in the section 3.

The X-ray CCD camera onboard ASTRO-H is called SXI[2], and covers the 0.5-12 keV energy band. The system has four (2x2 array) CCDs, each of which has 1280x1280 imaging pixels and is read out as the 640x640 format with the on-chip pixel binning technique. The data are digitized to 12 bits per pixel with a 4-bit attribute. As all of the digital data have to be acquired within an exposure cycle (4 sec), the data transfer speed of 8.8 Mbps (including margin) is required.

2 SXI BBM

The components of the breadboard model (BBM) [3] are shown in Figure 1. Sequencer supplies timing clocks to DE I/F, Driver Board and Video Board. Driver Board provides analogue signals for a CCD. Video Board performs the former-half of ΔΣ-digitization for the outputs of a CCD and send the bit stream out to DE I/F.

DE I/F was developed by Mitsubishi Heavy Industries Ltd. using the Universal SpW Board, which includes two FPGAs: User FPGA and SpW FPGA. User FPGA convolves the CCD bit stream to 12 bit values, attaches the 4 bit pixel codes (PCODEs) to them and stores them in SDRAM. PCODEs are supplied from Sequencer and have the pixel-type information such as active, inactive, frame start or line start. The 2 MB space is allocated for the frame data in the SDRAM. DE I/F is connected to a SpW network, and the SpW FPGA takes care of the communication. As the first step of the SpW-based communication, we used a SpaceCube1 (simply SpaceCube or SpC, hereafter) as the counterpart. The SpC is also connected to the POSIX OS computer by Ethernet.

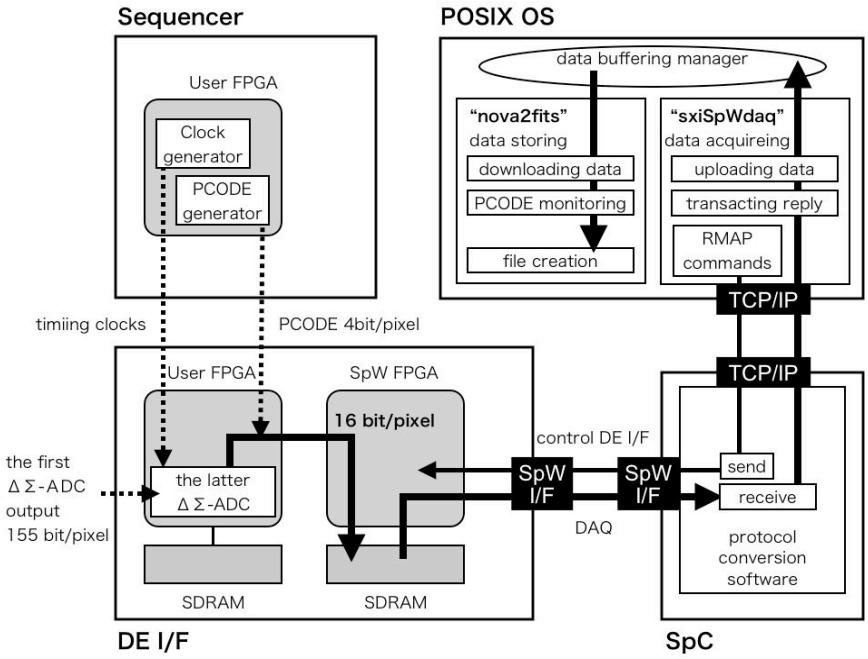


Figure 2: A schematic diagram of the developed DAQ system. We implemented the data acquiring software and the data storing software running on POSIX OS. The accesses to DE I/F are handled by RMAP communication library. Digitized data are acquired from the SDRAM attached to the SpW FPGA to the POSIX system.

3 DEVELOPMENT OF THE DATA ACQUISITION SYSTEM

In this section, we focus on the digital components of the BBM, which are drawn as the greyed boxes in Figure 1. Using the digital part of the BBM, we developed the X-ray CCD data acquisition system as shown in Figure 2.

3.1 DATA ACQUISITION STRATEGY

We set up the SpC as the protocol converter between SpW and TCP/IP; the conversion software is a part of "SpaceWire/RMAP Library" [4]. To retrieve the SDRAM data, we implemented the data acquisition software named "sxiSpWdaq", which uses the library and carries out the RMAP-related functions such as the packet formation and handling. This software repeats the following two steps: (1) sending the data-acquisition start command out to the User FPGA logic and (2) reading the stored data through SpW. The software receives 6 KB of data by each RMAP access, and sends them to internal data buffering manager in order to store them in a file system.

3.2 SPACEWIRE PERFORMANCE

After implementing the system, we carried out a test run over 24 hours, and the system continued working until stopped explicitly. This shows the stability of the SpW system. The transfer speed was, however, insufficient: the achieved mean speed was 4.4 Mbps, which is faster than the speed for 1 CCD but too slow for 4 CCDs. The bottle neck is the protocol conversion. To send SpW packets over TCP/IP, we have to send each packet with a header that acts as the packet delimiter and includes the packet length: thus the SpC has to receive an RMAP read command packet with two steps, reading the header and the body. It seems, unfortunately, that the TCP/IP driver on the SpC is not optimised for such bytes-to-bytes read

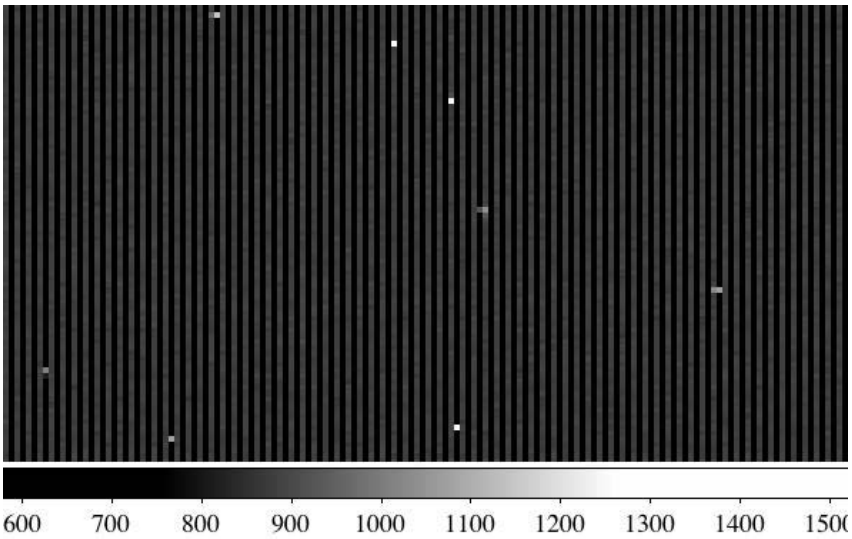


Figure 3: An example of a part of the frame image: we selected a part of the CCD chip. The X-ray events are seen as white dots.

call and the throughput decreases significantly.

3.3 DATA STORING

We also implemented the data storing software called "nova2fits", which obtains the acquired data from the data buffering manager. This software monitors the PCODE of each pixel, and creates an image file of the FITS format, which is the standard of astronomical electric data, with CCfits and CFITSIO libraries [5] when the last pixel of the image comes. This kind of the image reconstruction is required because the SDRAM data does not start from the beginning of a frame every time due to a skew between the starts of Sequencer clock and of User FPGA process of DE I/F. This software does not have a significant affect on the DAQ throughput because this runs on the POSIX machine with plenty of processing power.

4 DEMONSTRATION OF END-TO-END DATA ACQUISITION

With this system, we drove the whole BBM components including the analogue parts. We used a 512x608 CCD chip, cooled it to -15 °C in a vacuum chamber in order to reduce the thermal dark current, and irradiated X-rays of ^{55}Fe. The readout pixel rate is 100 kHz and no pixel binning was held. One of the obtained images is shown in Figure 4. We verified that the BBM adequately works as an X-ray CCD system.

5 REFERENCES

[1] Ozaki, M. et al., "SpaceWire driven architecture for the ASTRO-H satellite", this conference, 2010
[2] Tsunemi, H. et al., "The SXI: CCD camera onboard the NeXT mission", proceedings of SPIE Astronomical Instrumentation, 2008
[3] Anabuki, N. et al., "SpaceWire application for the X-ray CCD camera onboard the ASTRO-H satellite -The BBM development and the EM design-", this conference, 2010
[4] Yuasa, T. et al., "SpaceWire/RMAP-based Data Acquisition Framework for Scientific Instruments: Overview, Application, and Recent Updates", this conference, 2010
[5] NASA Goddard Space Flight Center, "FITSIO Homepage", http://heasarc.gsfc.nasa.gov

DATA PACKET FLOWS MULTIPLEXING IN SPACEWIRE ROUTING SWITCHES

Session: SpaceWire Networks and Protocols (Poster)

Short Paper

Alexander Glushkov, Ilja Alexeev

SIC "ELVEES", Moscow

E-mail: grisly@elvees.com, ilja@elvees.com

Elena Suvorova, Yuriy Sheynin

St. Petersburg State University of Aerospace Instrumentation
67, Bolshaya Morskaya st. 190 000, St. Petersburg RUSSIA

E-mail: sheynin@aanet.ru, suvorova@aanet.ru

ABSTRACT

In many applications SpaceWire networks are used for data transmission from sensors (sensor fields) to an onboard computer, for transmission from a computer to distributed actuators, data sinks. Multiplexing of data packet flows with small density to high density packet flow task and inverse task of a high-rate flow demultiplexing to multiple lower density flows arise. In the article we consider problems of packet flow multiplexing and demultiplexing with SpaceWire routing switches.

Balancing parameters of packet flows, link rates and buffer sizes governs efficiency of the buffering mode, routing mode. We consider dependency between utilization of the high speed link and buffering, buffer sizes on input and output ports, used in the system typical packet size, relation between link speed and data flow density in different links, evaluate relation between hardware cost of a routing switch hardware implementation and high speed link utilization.

1 INTRODUCTION

The main mode in the SpaceWire is wormhole routing with switching on-the-fly. Buffering of the packets is also possible. Packet forwarding through switch fabric starts when it is received in the buffer or the buffer is full; the packet size could be larger than the buffer size. The buffering mode in a routing switch could be used to conform speeds between input and output routing switch ports. It could improve output port and link utilization by multiplexing data flows, high-rate input port utilization by data outflows demultiplexing. Role of this mode illustrated by Figure 1.

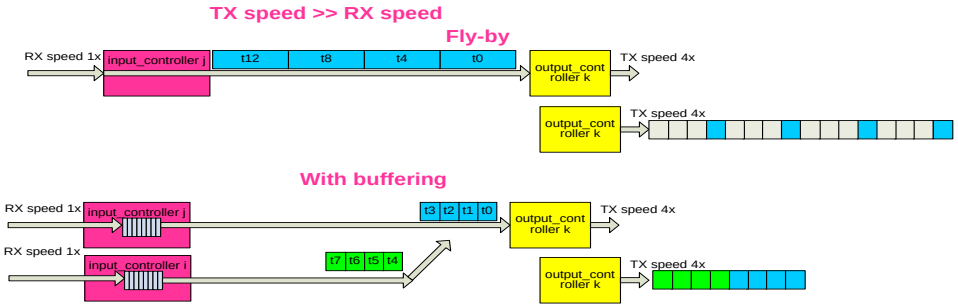


Figure 1

Let's evaluate packet transmission time via system that includes some source nodes, connected by switch to one host. For example, system includes 15 source nodes and 16-ports routing switch SpW. Data flows from nodes are equal and low intensive.

2 PACKET TRANSMISSION TIME EVALUATION

We consider next variants of system configuration (in switch are used dynamic cyclic priorities):

	Full buffering is sources	Full buffering in input ports of switch	Routing mode in switch
1	Not used	Not used	On the fly
2	used	Not used	On the fly
3	Not used	used	With full buffering
4	used	used	With full buffering

For configuration 1 maximal time of packet transmission is:

$$T_{\max 1} = T_{\text{rec}} + T_h + (N-1) * T_{trans} + \mathrm{T}_{\text{trans}} \qquad (1)$$

Trec – transmission time of the packet header from source node to input port of switch. It depends on link transmission rate.
Th – header processing time
Third component of this formula defines maximal output port waiting time
N – number of sources

Ttrans – transmission time to host. It is function from min(packet generation rate, link-from-source rate, link-to-host rate) and packet length, buffer size in input port (packet could collect in this buffer when it wait for translation to output port) and load of output port. In this example we suppose that minimal is packet generation rate.

For configuration 2 maximal time of packet transmission also could be evaluated by formula (1). But in this case Ttrans not depends on packet generation rate, because packet is fully buffered on the source

For configuration 3 maximal time of packet transmission is:

$$T_{\max 3} = T_{bufsw} + T_h + (N-1) * T_{trans} \qquad (2)$$

Tbufsw – time of collection packet in switch's input port buffer. It depends on data packet flow generation rate (we suppose that it is lower than link rate) and packet length

For configuration 4 maximal time of packet transmission is:

$$T_{\max 4} = T_{bufs} + T_{bufsw} + T_h + (N-1) * T_{trans} \quad (3)$$

where Tbufs – time of packet collection in source buffer (depends on packet generation rate)

Tbufsw – time of collection packet in switch's input port buffer. It depends on link rate and packet length.

Average packet transmission time for configuration 1 and 2 is:

$$T_{av1} = T_{rec} + T_h + W + T_{transav} = T_{rec} + T_h + T_{ransav} * \frac{R_{out}^2}{(1-R_{out})} + T_{transav} \quad (4)$$

where W – average time of waiting output port

Ttranav – average time of transmission packet to output port. Ttranav=Ttbuf+Ttin.

Ttbuf – transmission time of packet's part that placed in input port. Ttbuf=(W/Ttin1)*Tout1, where Ttin1 – transmission time of one symbol from host to switch's input port buffer, Tout1 – transmission time of one symbol via output port of switch to host.

Ttin – время передачи оставшейся части пакета из входного порта (со скоростью генерации данных в источнике для режима 1 или со скоростью передачи данных по линку для режима 2). Ttin=(Plen- W/Ttin1)*Ttin1

Rout – load of output port

$$\text{Rout} = (N-1) * (T_{\text{transav}}) * L_{in}$$

Lin –density of packet flow from one source. Lin=1/(Plen*Rs), where Plen – packet length, Rs data generation rate in source.

Average packet transmission time for configuration 3 is:

$$T_{av3} = T_{bufsw} + T_h + W + T_{transav} = T_{bufsw} + T_h + T_{ransav} * \frac{R_{out}^2}{(1-R_{out})} + T_{transav} \quad (5)$$

In difference from formula (4) in this case Ttransav=Plen*tout1

Average packet transmission time for configuration 4 is:

$$T_{\text{av4}} = T_{bufs} + T_{\text{bufsw}} + T_h + W + T_{transav} = T_{\text{bufsw}} + T_h + T_{ransav} * \frac{R_{out}^2}{(1 - R_{out})} + T_{transav} \quad (6)$$

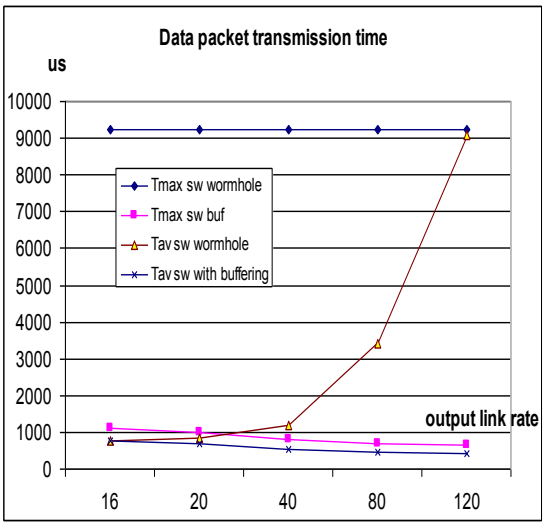
In this formula Ttransav derived as in (5)

3 RESULTS AND CONCLUSSION

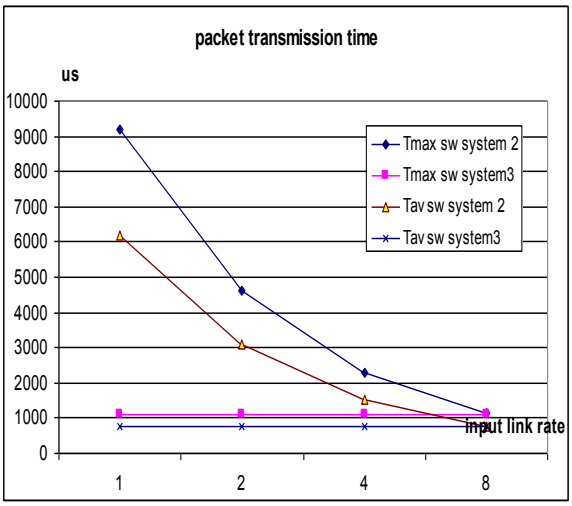
We compare timing parameters of four configurations in same switch (with same input buffer size) for transmission packets which length is not more than buffer size.

If throughput of output port/link is not essentially bigger, than summary data transmission rate from all sources then results for system with routing on the fly and for system with full buffering are practically equal (Figure 2, a). It happens because when we use routing on the fly first packet goes to output port very slowly (on packet generation rate or on link rate). On this time packets from other sources collected on buffers of input ports and then translated to output port very quickly (on output port link rate).

Timing parameters of configuration 3 are better that timing parameters of configuration 2 but if input link rate grow and throughput of output port/link is not essentially bigger, than summary data transmission rate from all sources parameters of these configurations are practically equal (Figure 2 ,b).



a)



b)

Figure 2

IMPLEMENTING THE CCSDS PACKET TRANSFER PROTOCOL FOR ECSS SPACEWIRE LINKS

Session: SpaceWire Standardisation (Poster)

Sandi Habinc, Marko Isomäki

Aeroflex Gaisler, Kungsgatan 12, SE-411 19 Göteborg, Sweden

sandi@gaisler.com marko@gaisler.com

ABSTRACT

Transmission of CCSDS Space Packets [4] over ECSS SpaceWire links [10] is nothing new, but with the new ECSS SpaceWire protocols standards, it is possible to implement telemetry encoders and telecommand decoders offering interoperability. There is however an alternative method also based on the ECSS SpaceWire protocol standards offering additional services. The two approaches are compared in this paper.

CCSDS PACKET TRANSFER PROTOCOL

1.1 OVERVIEW

As stated in the *SpaceWire - CCSDS packet transfer protocol*, ECSS-E-ST-50-53C standard [13], the *CCSDS Packet Transfer Protocol* (called CPTP hereafter) has been designed to encapsulate a CCSDS Space Packet into a SpaceWire packet, transfer it from an initiator to a target across a SpaceWire network, extract it from the SpaceWire packet and pass it to a target user application.

1.2 FORMATS

The CCSDS Space Packet defined in [4] is shown in the figure hereafter. The CCSDS Packet Transfer Protocol should not be confused with the Encapsulation Packet [7]

Space Packet									
Primary Header							Packet Data Field		
Packet Version Number	Packet Identification			Packet Sequence Control		Packet Data Length	Secondary Header (optional)	User Data Field	Packet Error Control (optional)
	Type	Secondary Header Flag	Application Process Id	Sequence Flags	Sequence Count				
0:2	3	4	5:15	16:17	18:31	32:47			
3 bits	1 bit	1 bit	11 bits	2 bits	14 bits	16 bits	variable	variable	variable

defined by CCSDS.

Figure: CCSDS Space Packet

The CCSDS Space Packet is one of several data units [5] that can be transferred in Telecommand [3][9], Telemetry [1][8] or AOS Transfer Frames [2]. The CCSDS Packet Transfer Protocol packet is shown in the figure hereafter.

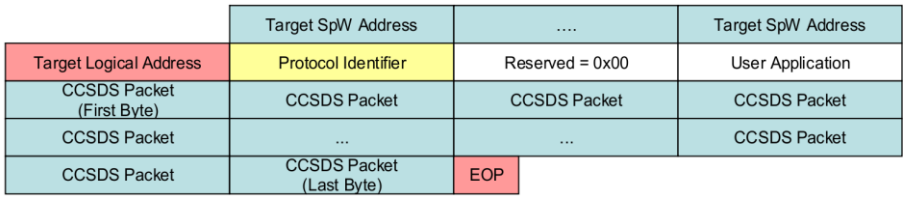


Figure: CCSDS Packet Transfer Protocol packet

1.3 SERVICES

The CPTP approach provides a unidirectional data transfer service from a single source user application to a single destination user application through a SpaceWire network. The protocol does not provide any means for ensuring delivery of the packet nor is it responsible for the contents of the packet being a CCSDS Space Packet. This actually opens up the possibility to transfer other types of data as per [5] than CCSDS Space Packets, e.g. SCPS-NP [6] or Encapsulation Packet [7], although not allowed.

The protocol does provide several steps of checking before a CCSDS Space Packet is passed to the target user application, of which one introduces some level of complication when implemented in hardware.

Implicitly it is assumed that the *Target Logical Address field* is matching the destination, and that the *Protocol Identifier field* is 0x02. Any mismatch should result in the received SpaceWire packet not being considered for the CPTP handling.

The *Reserved field* is checked to be all zero, if not then the received SpaceWire packet is discarded and an error indication is sent to the target application. The Reserved Field is located close to the beginning of the SpaceWire packet, making it easy to check without the need to buffer any part of the SpaceWire packet.

If the SpaceWire packet is completed with an End Of Packet (EOP) the CCSDS Space Packet is passed to the target user application. However, if SpaceWire packet is ended with an Error End of Packet (EEP), SpaceWire packet is discarded and an error indication is sent to the target application.

1.4 IMPLICATIONS

By definition, EOP or EEP are at the end of the SpaceWire packet, requiring the SpaceWire packet to be temporarily buffered before the check can be performed. With a CCSDS Space Packet size of maximum 65542 octets, this can be problem from an implementation point of view, especially if implemented completely in hardware.

CPTP does not provide any means for status reporting.

1.5 SIMILAR IMPLEMENTATIONS

The new CPTP protocol has some similarity with what is used in the Single Chip Telemetry and Telecommand device (SCTMTC or AT7909E) [15]. The *User Application field* could be seen as a mechanism for routing, as the SCTMTC uses the first byte of the SpaceWire packet header to route Space Packets to different telemetry Virtual Channels. The actual Space Packet is carried in the cargo of the SpaceWire packet, and is ended with an EOP. What is new for CPTP is the introduction of the

Target Logical Address, *Protocol Identifier* and *Reserved* fields, the two former being specified also for Remote Memory Access Protocol (RMAP) [12].

For the SCTMTC we implemented a mechanism for retracting a Space Packet that had been partially inserted in the telemetry encoder in the case the reception of a SpaceWire packet was terminated with an EEP. This is similar to what is required by CPTP. The buffering of the Space Packet was thus done in the telemetry encoder, which is implemented for nominal functionality and did not require extra resources.

REMOTE MEMORY ACCESS PROTOCOL

1.6 OVERVIEW

The SpaceWire Remote Memory Access Protocol (RMAP) [12] allows the implementation of a standardized communication method for reading and writing to remote memory and registers. This eliminates the plethora of existing ad hoc protocols that have been used in previous developments, allowing designers to focus their efforts on a single re-usable solution that can be transferred between projects.

1.7 FORMAT

	Target SpW Address	....	Target SpW Address
Target Logical Address	Protocol Identifier	Instruction	Key
Reply Address	Reply Address	Reply Address	Reply Address
Reply Address	Reply Address	Reply Address	Reply Address
Reply Address	Reply Address	Reply Address	Reply Address
Initiator Logical Address	Transaction Identifier (MS)	Transaction Identifier (LS)	Extended Address
Address (MS)	Address	Address	Address (LS)
Data Length (MS)	Data Length	Data Length (LS)	Header CRC
Data	Data	Data	Data
Data	...	...	Data
Data	Data CRC	EOP	

The RMAP format is defined in [12] and a write command is shown hereafter.

Figure: RMAP Write Command

The adoption of the Advanced Microcontroller Bus Architecture (AMBA) [14] as the on-chip bus fabric used in ESA developments was made simultaneously with the development of the LEON processor. The combination of RMAP and AMBA provides a means for remote access via SpaceWire to resources in a system-on-chip design, any AMBA slave connected to the bus can thus be read and written to. This is utilized in an alternative method for sending CCSDS Space Packets over SpaceWire.

1.8 SERVICES

Aeroflex Gaisler has implemented telemetry encoders and telecommand decoders in Field Programmable Gate Array (FPGA) devices [16] where the communication is done via SpaceWire links, but by means of the RMAP rather than CPTP.

The Space Packet is carried in the *Data field* of a RMAP write command. The RMAP protocol provides protection of the Space Packet by means of the 8-bit *Data CRC field*, which can be used to discard any packets that have been received with errors. RMAP also supports acknowledgement reporting to the initiator. (Space Packets can themselves include a 16-bit CRC as optional Packet Error Control, but would require checking of the Space Packet which is not in line with a layered protocol approach.) The routing is done by means of the addressing capability of RMAP commands; e.g. the address can be used for distinguishing between virtual channels on a downlink.

The Space Packet is written into an AMBA slave, which is addressed over the AMBA bus using the RMAP *Address field*. The AMBA slave forms the input to Virtual Channel Generation function of a telemetry encoder. Space Packets or any other user-defined data block can be input. Writing is only possible when the packet valid delimiter is asserted, else the access results in an AMBA access error. In the case the data from a previous write access has not been fully transferred over the interface, a new write access will result in an AMBA retry response. The progress of the interface can be monitored via the AMBA bus, which incorporates status and monitoring functions including busy and ready signaling for a new word or a new Space Packet.

COMPARISON

Since both RMAP and CPTP adhere to the same SpaceWire protocol identification ECSS standard [11], there is no problem mixing them in the same implementation. What they have in common is that a SpaceWire packet carries one-and-only-one Space Packet.

The CPTP protocol does however neither provide means for reporting the delivery of the packet, nor adding data error detection as with the RMAP approach (other than parity on the SpaceWire link, which both approaches implement). Consequently CPTP requires fewer overheads.

CONCLUSIONS

Both RMAP and CPTP provide basics means for CCSDS Space Packet transfer over SpaceWire, the former providing more protection against errors and possibility for acknowledgement and status monitoring, whilst the latter results in less overhead.

REFERENCES

1. TM Space Data Link Protocol, CCSDS 132.0-B-1, www.ccdsds.org
2. AOS Space Data Link Protocol, CCSDS 732.0-B-2
3. TC Space Data Link Protocol, CCSDS 232.0-B-1
4. Space Packet Protocol, CCSDS 133.0-B-1
5. Space Link Identifiers, CCSDS 135.0-B-4
6. Space Communications Protocol Specification (SCPS) - Network Protocol (SCPS-NP), CCSDS 713.0-B-1

7. Encapsulation Service, CCSDS 133.1-B-2
8. Telemetry transfer frame protocol, ECSS-E-50-03A, www.ecss.nl
9. Telecommand protocols, synchronization and channel coding, ECSS-E-50-04A
10. SpaceWire - Links, Nodes, Routers and Networks, ECSS-E-ST-50-12C
11. SpaceWire protocol identification, ECSS-E-ST-50-51C
12. SpaceWire - Remote memory access protocol, ECSS-E-ST-50-52C
13. SpaceWire - CCSDS packet transfer protocol, ECSS-E-ST-50-53C
14. AMBA Specification, Rev 2.0, ARM IHI 0011A, Issue A, ARM Limited, www.arm.com
15. Single Chip Telemetry and Telecommand, AT7909E, Data Sheet, Atmel, www.atmel.com
16. Sandi Habinc et al., *"ECSS TM/TC Component with SpaceWire RMAP Interfaces"*, International SpaceWire Conference, 2010

IMPLEMENTING SPACEWIRE RMAP LINKS IN FLASH-BASED FPGA TECHNOLOGY

Session: SpaceWire Components (Poster)

Short paper

Sandi Habinc, Jonas Ekergarn, Kristoffer Glembo

Aeroflex Gaisler, Kungsgatan 12, SE-411 19 Göteborg, Sweden

sandi@gaisler.com, ekergarn@gaisler.com, kristoffer@gaisler.com

ABSTRACT

Aeroflex Gaisler (Sweden) is developing the software and the digital part of a Motion Control Chip (MCC) in an activity that is lead by ÅAC Microtec (Sweden) under a contract with the European Space Agency (ESA). The digital logic is implemented in a Flash-based FPGA, including SpaceWire [1] RMAP [2] interfaces.

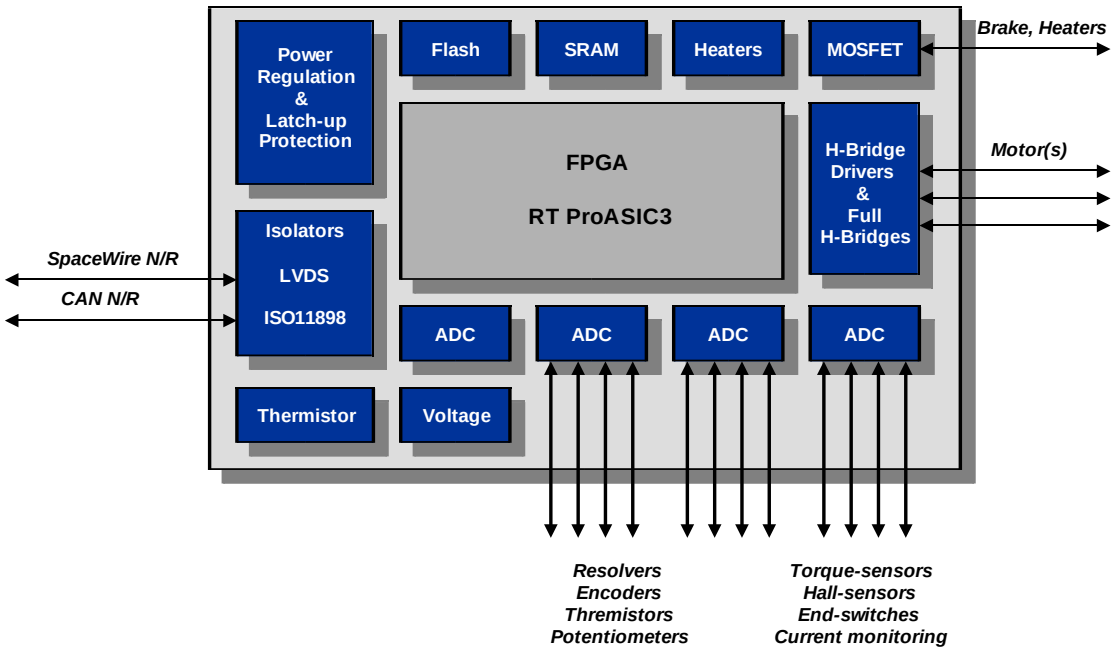
OVERVIEW

The Motion Control Chip (MCC) is a freestanding component that can control up to three brushed motors or one brush-less motor in torque, position or velocity mode, and will be implemented as an advanced 3D-multi-chip-module (3D-MCM) [3].

The baseline design includes a field programmable gate array (FPGA) as a naked die for the implementation of the digital part. To allow for programmability and future enhancements, a re-programmable FPGA has been selected.

The choice of a re-programmable over a one-time programmable FPGA has been driven by various factors, the two most prominent being that the 3D-MCM should be possible to program to different customer's needs and that qualification of programming procedures for one-time programmable FPGAs as a naked die is not straightforward.

Figure: Block diagram of MCC



FUNCTIONALITY OF FPGA

1.1 OVERVIEW

The digital design forms a system-on-a-chip comprising key elements such as the fault-tolerant LEON3-FT 32-bit SPARC processor [5], optional floating-point unit, debug support unit, etc. The main interfaces of the FPGA are listed hereafter:

- SpaceWire link with optional RMAP to support remote memory access for software download and debug, based on ECSS standards
- Optional redundant CAN 2.0A/B bus interface, based on ECSS standards
- SPI interface for access to ADC devices, support for multiple accesses in parallel to allow correlations
- Pulse Width Modulation: symmetric and asymmetric
- General Purpose Input Output
- Memory Controller with EDAC to protect external PROM & SRAM memory
- JTAG Debug Link, used for software download & debug

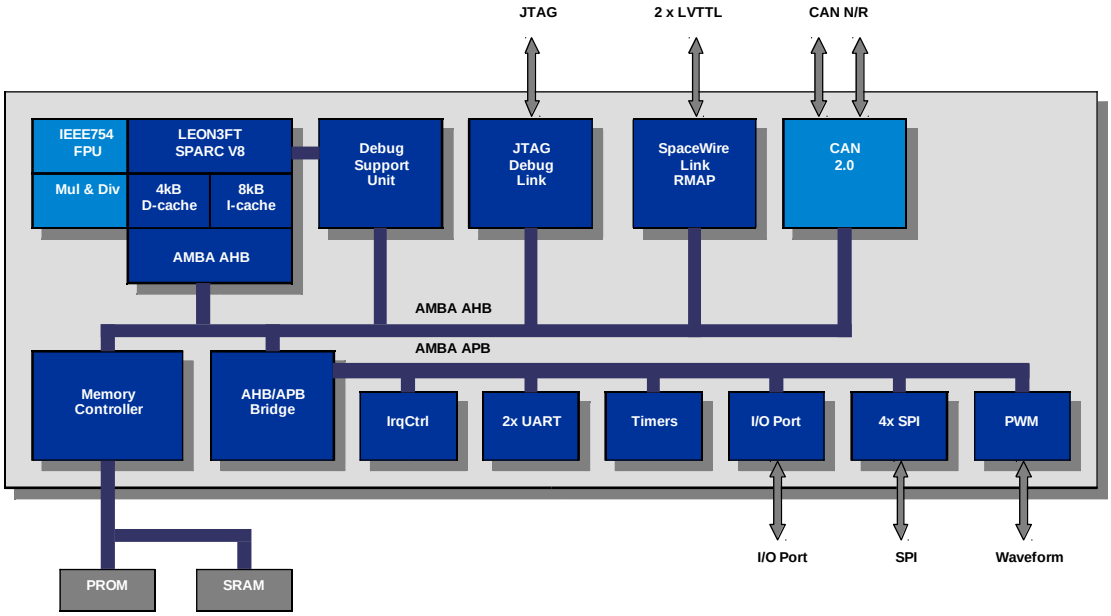


Figure: MCC FPGA block diagram

1.2 FPGA TECHNOLOGY

The Actel ProASIC3 RT3PE3000L FPGA part has been chosen for the implementation of the digital logic. Although this part can tolerate a total ionizing dose (TID) of up to 20 krad and is basically single event latch-up (SEL) free, it exhibits some sensitivity to single event upsets (SEU) and transients (SET) that needs to be taken into account during logical design.

Main features of the Actel RT3PE3000L FPGA are listed hereafter:

- 3,000,000 System Gates
- 75,264 Logic Tiles
- 504 kbits RAM
- 1 kbits FlashROM (user accessible)
- RT ProASIC3 use same process as commercial UMC 0.13 µm ProASIC3EL
- RT3PE3000L is the same silicon as the A3PE3000L

1.3 IMPLEMENTATION APPROACH

The MCC FPGA has been designed using the same IP cores that are used by Aeroflex Gaisler in their LEON3FT-RTAX product line that is based on Actel's RTAX2000S anti-fuse FPGA parts. The portability of the IP cores, all written in VHDL, permits a move from anti-fuse to Flash based FPGA technology with a minimum of effort [4].

The on-chip block memories have been protected against SEUs using either error correction and detection (EDAC) or simple parity code for the detection of bit errors. The flip-flops have been protected against SEUs using triple modular redundancy (TMR) that can be automatically inserted by the VHDL synthesis tool.

The probability of SETs in combinatorial logic at modest clock frequencies (25 MHz) is very low, according to reports from Actel, which should not require any mitigation but will be further investigated. SETs on I/Os and clock lines are monitored by dedicated logic that is checking for potential I/O bank turn-offs and detections lead to a processor interruption or a reboot.

A system level watchdog is used to cover any undetected effects due to SETs.

1.4 IMPLEMENTATION RESULTS

The MCC design (without CAN interfaces) gives the following approximate utilization results after synthesis and place&route results (RT3PE3000L -1):

- Size: > 95% (with TMR and EDAC)
- System frequency: 25 MHz

The following performance figures were obtained:

- CPU: 20 Dhrystone MIPS
- FPU: 4 MFLOPS
- SpaceWire: 20 Mbit/s (twice the requirement)
- CAN: 1 Mbit/s
- SPI: 10 Mbit/s
- JTAG: 1 Mbit/s

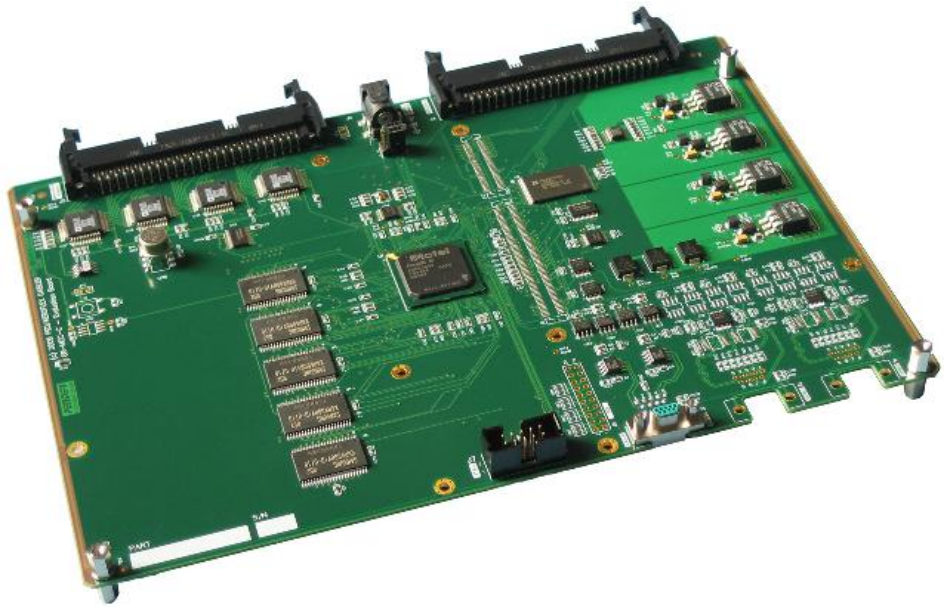


Figure: MCC-C FPGA development board

COMMUNICATION LINK IMPLEMENTATION

The system can be interfaced either via CAN or SpaceWire. The advantage of using SpaceWire links with built-in RMAP target capability is that the system can be controlled remotely, allowing upload of software to the non-volatile Flash PROM memory or directly to the volatile SRAM memory etc.

For dextrous robotic arms, wheels, masts, drills and functions needed on rovers, the approach is to use a selectively redundant CAN interface. This interface can be replaced with a single SpaceWire core without RMAP support, but with two external ports implementing automatic selective redundancy similar to the CAN approach.

For exoskeletons and applications where multiple motors are used, it is foreseen to use multiple SpaceWire links per unit that are interconnected through software or hardware routing. This allows reduction in harness for elongated structures etc. Note that the optional floating-point-unit would not fit in these configurations. There are two ways in which the links are implemented, either using two SpaceWire cores without RMAP and do the routing in software, or using a newly developed SpaceWire router [6] with two external ports and one internal port with DMA and RMAP support. The latter approach is to be implemented and evaluated in the future.

To allow flexibility and fault-containment, the Low-Voltage Differential Signaling (LVDS) buffers required for SpaceWire are implemented with off-chip parts.

CONCLUSIONS

The re-programmable RT ProASIC3 FPGA technology fits well within applications with moderate radiation requirements. The in-situ programmability enables the development of highly miniaturized systems which can be adapted to customers needs late in the development cycle. Porting a LEON3-FT system with SpaceWire links from anti-fuse to Flash-based technology went smoothly, with much of the work already performed previously for the commercial version of the IP core library.

ACKNOWLEDGEMENTS

The authors acknowledge ESA for the commissioning and funding of the development of the Motion Control Chip under ESA contract number 21737, lead by Dr. Johan Köhler. The authors also acknowledge ÅAC Microtec as the prime of the Motion Control Chip development activity.

REFERENCES

1. SpaceWire - Links, Nodes, Routers and Networks, ECSS-E-ST-50-12C
2. SpaceWire - Remote memory access protocol, ECSS-E-ST-50-52C
3. F. Bruhn et al, *"Low Temperature Miniaturized Motion Control Chip - Enabled by MEMS and Microelectronics"*, ASTRA 2008
4. S. Habinc et al., *"Using a FLASH Based FPGA in a Miniaturized Motion Control Chip"*, MAPLD 2009
5. S. Habinc et al., *"Leveraging the Availability of 32-bit Fault-Tolerant Processors Suitable for Radiation-Tolerant FPGA Devices"*, CMSE 2010
6. M. Isomäki et al., *"A Configurable SpaceWire Router VHDL IP core"*, ISC 2010

SPACE CUBE 2 SOFTWARE DESIGN KIT (SDK)

Session: SpaceWire Test and Verification (Poster)

Short Paper

Hiroki Hihara, Masashi Uo, and Masaaki Iwasaki

NEC TOSHIBA Space Systems, Ltd., 10, Nisshin-cho 1-chome, Fuchu, Tokyo, Japan

Toru Tamura, and Shuichi Moriyama

NEC Soft, Ltd., 2-2-41, Ekimae, Kashiwazaki, Niigata, Japan

Naoki Ishihama

JAXA's Engineering Digital Innovation Center, Japan Aerospace Exploration Agency (JAXA), 2-1-1 Sengen, Tsukuba, Ibaraki, 305-8505, Japan

Masaharu Nomachi

Laboratory of Nuclear Studies, Graduate School of Science, Osaka University,

1-1 Machikaneyama, Toyonaka, Osaka 560-0043

Tadayuki Takahashi and Takariho Yamada

Department of High Energy Astrophysics, Institute of Space and Astronautical Science (ISAS), Japan Aerospace Exploration Agency (JAXA), 3-1-1 Yoshino-dai, Chuoh-ku, Sagamihara, Kanagawa 252-5210, Japan

E-mail: h-hihara@bc.jp.nec.com, m-uo@bq.jp.nec.com, m-iwasaki@kd.jp.nec.com, tamura@mxm.nes.nec.co.jp, moriyama@mxp.nes.nec.co.jp, ishihama.naoki@jaxa.jp, nomachi@lns.sci.osaka-u.ac.jp, takahasi@astro.isas.jaxa.jp, tyamada@pub.isas.jaxa.jp

ABSTRACT

Space Cube2 is a core control node in medium size satellites as ASTRO-H as well as our small satellites, which are based on SpaceWire communication networks. In order to provide software design kit (SDK) for Space Cube2, reference computer architecture, communication model, and database scheme for satellite operation are defined. Reliability and timeliness characteristics are also taken account while establishing communication protocol stack. The SDK consists of T-Kernel real-time operating system (RTOS), middleware, software tools, and hardware support equipment. Single source code of T-Kernel is provided as open source, and standard middleware is developed by Japanese space industries and JAXA/ISAS.

1 SPACE CUBE2 DESIGN FRAMEWORK WITH SPACEWIRE

Space Cube2 shown on figure one is used for several subsystems as data handling subsystem (DHS), attitude and orbit control subsystem (AOCS), and mission data processor (MDP) as a general purpose onboard computer. Since it is used over wide range of programming style of each subsystem, a reference design framework is

required for sharing the tools and methods used during application development processes.

Figure 1 Space Cube2

It is practical way to establish design framework as a software development kit (SDK) in order to make developers understand the reference model of software and hardware of the onboard computer. The reference model includes computer architecture, communication model, and database scheme for satellite operation. Space Cube architecture [1] is a reference for Space Cube 2, which defines the required specification of general purpose onboard computers. Communication model consists of Functional Model of Spacecraft (FMS) [2] and Spacecraft Monitor & Control Protocol (SMCP), [3] which are defined by JAXA/ISAS. Operation procedure of satellite is managed through Definition of Spacecraft Information Base 2 (DSIB2) [4], which is also defined by JAXA/ISAS. How to use middleware application program interface (API) is also provided as documents. In order to establish the design framework, several satellites as the earth observation satellites, LEO scientific satellites and inter-planetary scientific satellites have been investigated. Based on the assessment, standard middle ware requirement, telemetry / command design criteria, and network design criteria have been published.

1.1 Space Cube Architecture

Space Cube Architecture is established for developing Space Cube2. The architecture features following requirement;

1) Space Cube Architecture is derived from T-Engine architecture. T-Engine is an open platform for embedded use, which is applicable for various kinds of microprocessors.

2) SpaceWire is a mandatory interface for realising scalable network based on spacecraft architecture.

3) Compatibility is maintained through the standard middleware specification and API in order to enable accommodating various microprocessors.

4) In order to satisfy small size, light weight, low power consumption and low cost requirement for small satellites, one-chip microprocessors with its peripheral I/O devices embedded in itself are recommended.

1.2 Functional Model of Spacecraft (FMS) and System Information Base (SIB2)

FMS is defined through object oriented analysis in addition to the operation experience of ISAS satellites over many years. FMS defined through the conception,

which consists of attribute, operation, event, alert, behaviour, and diagnostic rule. The schema is defined in SIB2. The definition of SIB2 is published from ISAS and user interface is provided as spread sheet.

1.3 SPACECRAFT MONITOR & CONTROL PROTOCOL (SMCP)

The standard middleware is based on Space Monitor & Control Protocol (SMCP). SMCP has been developed by JAXA/ISAS, which aims at unified building method of commands, telemetry messages, and sequence for all satellites and onboard equipments.

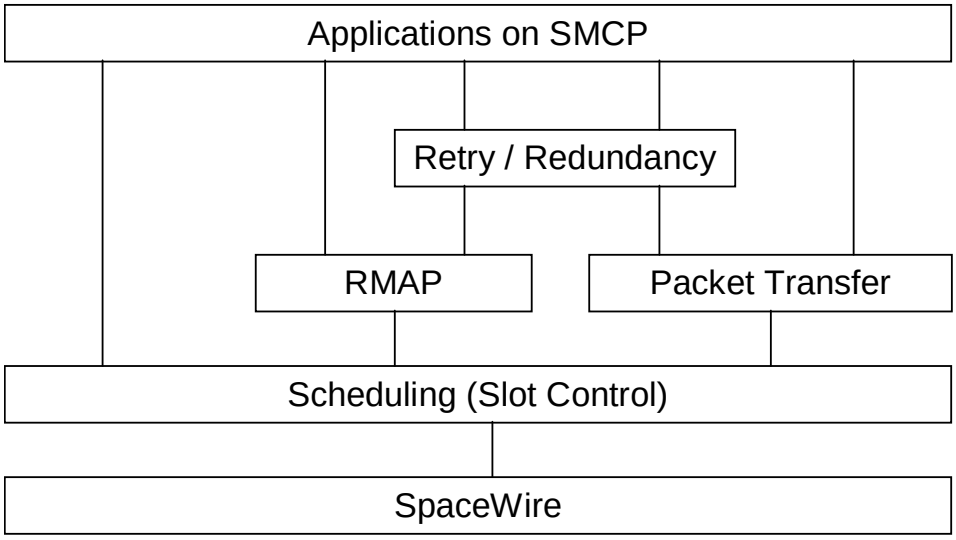


Figure 2 SMCP and protocol stack

Figure 2 shows the application development scheme. Telemetry and Command processing functions are realised through SMCP.

Reliability and timeliness are taken into account by exploiting RMAP and Time-Code capability. Retry and Redundancy control are carried out using CRC in RMAP packet. Scheduling (Slot Control) are implied by Time-Code. Initiator node should know the time slot, where as Target node doesn't have to know the time slot provided that the node can respond to RMAP packets.

2 TOOLS

Dedicated complier for HR5000 radiation hardened processor is developed by JAXA. Various software tools and equipments, which include low cost In-Circuit Emulator (ICE) for HR5000, are developed by commercial industries as well as space system industries. These tools are integrated on Eclipse framework and commercial version of SpaceCube2 is provided for desk-top space-use program development.

This SDK is developed among space system community which includes universities, so multi-lingual text books are also provided for widening space system community.

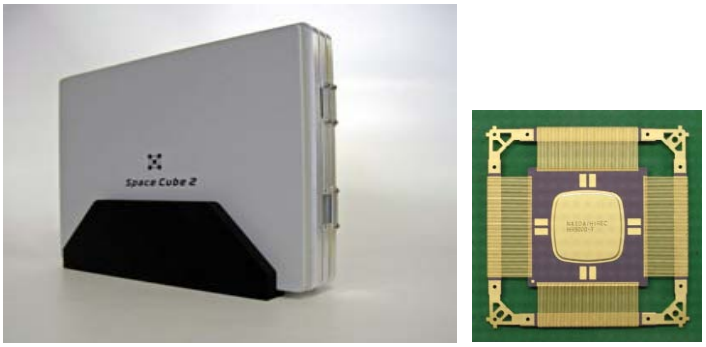


Figure 3 Commercial level version of Space Cube2 (with HR5000 processor)

Figure 4 Space Cube (original version) with SpaceWire CUBA Software

Figure 5 Space Cube with Giga-bit Ethernet

(Space Cube is developed by JAXA/ISAS and Shimafuji Electric, Ltd.)

Figure 6 SpaceWire Router (BBM/14ports)

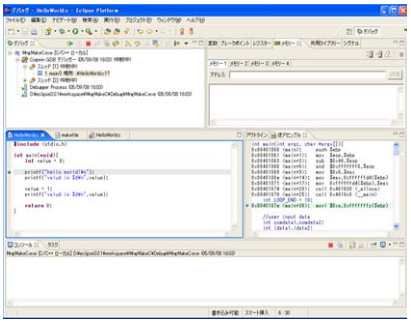


Figure 7 Integrated tools with Eclipse

3 REFERENCES

1. Tadayuki Takahashi, Takeshi Takashima, Satoshi Kuboyama, Masaharu Nomachi, Yasumasa Kasaba, Takayuki Tohma, Hiroki Hihara, Shuichi Moriyama, Toru Tamura, “SpaceCube 2 -- An Onboard Computer Based on SpaceCube Architecture”, International SpaceWire Conference 2007, 17-19 September 2007, p.65-68.

2. Takahiro Yamada, “Functional Model of Spacecraft (FMS)”, GSTOS 201, 15 September 2009.

3. Takahiro Yamada, “Spacecraft Monitor & Control Protocol (SMCP)”, GSTOS 200, 15 September 2009.

4. Takahiro Yamada and Keiichi Matsuzaki, “Definition of Spacecraft Information Base 2 (DSIB2), 15 September 2009.

LOW-MASS SPACEWIRE

Session: SpaceWire Components (Poster)

Short Paper

Jørgen Ilstad, Martin Suess

European Space Technology Centre, Noordwijk, Netherlands

E-mail: jorgen.ilstad(æ)esa.int, martin.suess(æ)esa.int

ABSTRACT

The current cable specification given in the ECSS-E-ST-50-12C SpaceWire standard [1] is defining the detailed construction of the cable. With this, the manufacturer can produce a cable compliant to the standard which is able to transmit the signal over a length of 10 m and support a data rate of 200 Mbps. A disadvantage is that this cable may be heavier and more rigid than necessary for short connections and too lossy for distances beyond 10 m.

For the upcoming update of the SpaceWire standard, the intention is not to specify the cable construction, but rather specify physical and electrical parameters which can be verified by measurement. These could comprise parameters like Differential Impedance, Signal Skew, Return Loss, Insertion Loss, Near-end Crosstalk (NEXT), Far-end Crosstalk (FEXT) and radiated EMI. From this specification, cable manufactures will be able to design SpaceWire compliant cables optimised for a particular application.

An ESA funded activity aimed to develop a low-mass SpaceWire cable was initiated at the beginning of 2010. The activity will define the electrical and physical performance parameters of the current SpaceWire cable and use these metrics to design a new cable with lower mass properties. Following this initial definition, ESA will develop, manufacture and validate a low-mass SpaceWire cable, a cable that is mainly targeted at shorter SpW interconnections. The goal is to reduce the mass by 30% to 50%. An important part of the activity is to determine whether the current SpaceWire cable grounding and shielding scheme can be changed without affecting performance factors such as EMC/EMI, mechanical properties and data rate adversely.

In this paper, the latest results obtained during the first part of the Low-Mass SpaceWire activity will be reported. It discusses candidate cable constructions to achieve lower mass, alternative connectors for the cable assembly and the upcoming tasks leading up to activity completion.

1 EFFECTIVENESS OF INNERSHIELDS

The SpaceWire cable uses multiple shields where the outer shield is connected to the spacecraft structure through the connector backshell, and the inner shields of the transmitting pairs is grounded to pin 3 of the MDM connectors. As shown in figure 1,

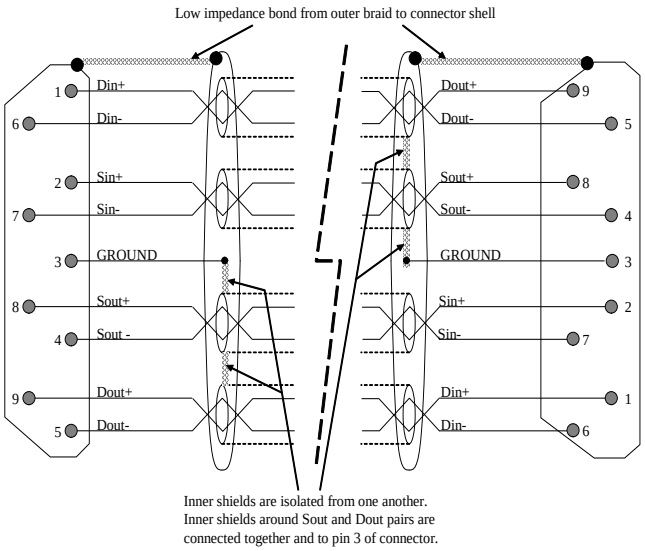


Figure 1: SpaceWire cable shield arrangement.

the inner shields are terminated at one end with the idea to avoid ground loops to occur. This type of arrangement is effective for cancelling out EMI at lower frequencies, but ineffective for higher frequency ranges. The rise and fall times of a typical LVDS signal is close to 1ns, which translates to a signal bandwidth in the range of 1-2 GHz. The notion that high frequency EMI is not effectively reduced by the inner shields, means that cross talk becomes an increasing problem both with respect to cable length and transmission rate. Tests carried out in [3] and [4] confirmed the inner shields to have little effectiveness.

2 SPACE CRAFT GROUNDING

In order to evaluate what parts of the current SpaceWire cable can be changed it is necessary to have an understanding of the typical space craft grounding practises. Although these may vary quite a bit from one mission to the other, there are commonalities which can be exploited. While diving into the complex domain of space craft grounding is out of scope in this paper, the findings in the initial phase of the Low-Mass SpaceWire activity shows that the inner shields of the SpaceWire cable could be connected on both sides without affecting its general use in a spacecraft.

3 REDUCING CABLE MASS

The previous sections highlighted the inner shields as not performing their intended function. Hence the cable construction could be adapted in a way which also reduces cable mass. Feasible solutions are to either remove the inner shields all together, while keeping the outer shield, - or removing the outer shield and grounding the inner shields at both ends to the chassis ground. To ensure good EMC performance it is important that these screens are continuous and are all 360˚ in contact with the backshell of the connector.

Considering the approaches which save the most mass, we first look the implications of removing the inner shields completely. The reduction of mass will give a 35-40% decrease in mass compared to the standard SpaceWire cable. Advantages of this approach are simpler termination of the shield to the connector backshell, and using 9-MDM for connections through bulkhead is possible. The drawback is the increased cross talk between the individual pairs. For lower data rates below 100Mbps and shorter cable lengths, this may not be a significant problem. The cross talk can to some extent be reduced by the twisting scheme of the individual pairs, however excessive cable tension or bend increases susceptibility to inter pair cross talk.

A second approach is to remove the outer shield completely and terminating the inner shields 360˚at each end to chassis ground. The termination of the shields is more mechanically challenging, but cable mass is significantly reduced and the cable will be much more flexible. With this approach the inter pair crosstalk is reduced and 9-pin MDM connectors will support bulkhead connections.

3.1 MATERIAL REDUCTIONS

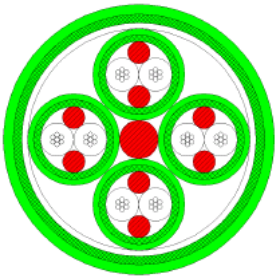


Figure 2: Suppression of fillers

Other approaches which further reduce mass is using lighter materials e.g. using silver plated aluminium instead of silver plated copper for the shields. Reducing the wire gauge is also an option which is viable for shorter cable lengths. In picture 2, the suppression of filler materials (in red) is considered to reduce mass. Adding all these options together leads to a potential mass reduction of around ~ 45 – 50 %. At present, the ongoing ESA study with Axon cable and University of Dundee has identified a range of candidates for mass reduction of the cable. Two candidate solutions will be the pre-selected for further analysis and tests. The final product will be subjected both to electrical characterisation, and mechanical stress tests which are compared to the performance of the current SpaceWire cable design.

3.2 ALTERNATIVE CABLE CONSTRUCTIONS

In the various options of different cable constructions, the well known twisted pair is usually selected. In addition, other alternative concepts based on thin coaxial cables are investigated carefully. A circular cable construction solution based on very thin coaxial cables is a promising candidate due to the inherent electrical performance. Such a design can carry very high data rates, up to several Gbit/s, with the appropriate connector for the cable assembly. It is estimated that using small conductor (AWG3401) gauge for the coax cables is estimated to give a potential weight saving in the range of 60% - 70%. A driving requirement is that such an option must be backwards compatible with 9-pin MDM connectors. The challenge is to ensure good mechanical robustness of the shield bonding towards connector.

4 ALTERNATIVE CONNECTORS

It may be attractive to also identify a small set of alternative connectors to be included in a future revision of the SpaceWire standard. These may serve applications where very high data rate is required, or where miniaturisation is needed. In the course of the Low-Mass SpaceWire (LMSPW) activity, one candidate connector that will be tested is the Nano-D connector which is most suited for application where miniaturisation is a key driver. An evaluation programme announced in ESA AO/1-6126 for this particular connector family is running in parallel to the LMSPW activity. The objective is to evaluate the family of ultra miniature Nano-D connectors with respect to the specific requirements of space applications or special space programs such as the Mars surface explorer. Connector samples manufactured as output of the parallel evaluation activity will be used in the frame of the LMSPW activity for evaluation of complete cable assemblies.

Impedance matched connectors are also under the microscope in the ongoing LMSPW activity. Previous work performed in [3], [6] have identified candidates which will be included in the assessment. Here miniaturisation may not be the driving requirement, but rather electrical performance, robustness, suitability for bulkhead connections and handling.

5 PROPOSED SPACEWIRE STANDARD REVISION

As mentioned in previous publications [5], the current SpaceWire standard contains a detailed specification of how the cable should be constructed. In the revision of the standard, it is foreseen to rather specify the electrical parameters and tolerances in order to allow for manufacturing of cables that suite a particular application rather than "one size fits all". A proposal for standardisation will be one of the outcomes from the LMSPW.

6 CONCLUSION

In this paper, the status of the currently ongoing activity for the development of a Low-Mass SpaceWire cable has been presented. Various elements of cable construction have been discussed with emphasis on proposed cable construction changes that can significantly reduce mass. The various options will be down-selected based on simulations. The remaining candidate solutions will be manufactured and assembled in various configurations, including different connector types and cable assembly lengths. Each sample is electrically characterised, both in terms of performance and EMC/EMI properties with the current ECSS-E-ST-50-12C cable specification. In addition, a set of mechanical endurance tests will be performed to validate mechanical robustness of the cable, specifically connector/cable junction.

For shorter cable runs in the range of up to 3-5 meters, the preliminary analysis performed in the frame of the LMSPW activity show promising results in terms of reducing the cable mass significantly.

Relaxing the cable construction requirements by replacing them with electrical performance parameters will leave more freedom to use cables which are tailored to specific applications. They can be for example optimised for low-mass, link distance, mechanical handling properties or cost and still fulfil the electrical requirements to guarantee the very low bit error rate which is characteristic for SpaceWire.

7 REFERENCES

1. "SpaceWire: Links, nodes, routers and networks", ECSS-E-ST-50-12C, 31 July 2008.

2. S. Parks, "Analysis of Grounding and Shielding Arrangements", ESA C23029, 2010

3. S. Allen, "SpaceWire cable and connector variations", ISC 2007

4. S. Allen, "SpaceWire Physical Layer Issues", MAPLD 2006

5. M. Suess, S. Parks, P. Crawford " Spacewire Cable Characterisation", ISC 2007

6. D. Schierlmann, P. Jaffe, “SpaceWire cabling in an operationally responsive space environment”

A VERSATILE SPACEWIRE CODEC VHDL IP CORE

Session: SpaceWire Components (Poster)

Short Paper

Marko Isomäki, Sandi Habinc, Jiri Gaisler

Aeroflex Gaisler, Kungsgatan 12, SE-411 19 Göteborg, Sweden

marko@gaisler.com, sandi@gaisler.com, jiri@gaisler.com

ABSTRACT

Aeroflex Gaisler is specialized in developing IP cores and System-on-chip solutions based on the AMBA [1] on-chip bus. A library of IP cores called GRLIB [2] is provided where two SpaceWire IP cores, the GRSPW and GRSPW2, are available for AMBA bus based systems providing a master interface, advanced DMA and a RMAP [3] target.

There is a need for a more basic SpaceWire core with a simpler host interface and this has been addressed with the new SpaceWire Codec IP core. The core has the same SpaceWire link interface as the GRSPW2 but a simplified FIFO based host interface.

This paper presents the requirements driving the design of the core and features of the final core.

INTRODUCTION

Aeroflex Gaisler has been providing SpaceWire interfaces intended for AMBA on-chip bus based systems for several years. Two IP-cores have been available, the GRSPW and GRSPW2, which contain advanced functionality such as DMA channels and an RMAP target. It has become evident from several projects that this functionality is not needed in many cases and the inherent area overhead narrows the range of suitable technology alternatives in these cases.

For example, there might be only one hardware unit requiring SpaceWire in a system and it does not have an AMBA AHB on-chip bus. In that case it would more efficient to let it access data directly to/from the transmitter and receiver FIFOs for example, when implementing router functions [4] or additional SpaceWire protocols such as the CCSDS packet transfer protocol [7].

To provide a core suitable for these applications the development of a stand alone SpaceWire codec IP core was started.

This paper will first go through the basic properties of the core followed by a deeper analysis of the external SpaceWire interface, the dual ports and finally make a comparison with existing SpaceWire codecs.

BASIC PROPERTIES

Most parts of the SpaceWire codec are identical to the encoder-decoder used in the GRSPW2 core. This has the benefit of large parts of the core already being verified, validated and proven in hardware. The difference is in the host interface. Where the

GRSPW2 has an AMBA based DMA engine and RMAP target processing the incoming data the SpaceWire codec instead has FIFOs in both the transmit and receive directions containing the SpW characters including the control bit to be able to differentiate between EOP/EEP and normal characters [2]. This makes for a significant area reduction and simplification of the host interface. Since the user has full-control of the order and timing of the characters it gives more flexibility on packet generation compared to a bus based DMA solution where the implementation of the SpaceWire core restricts what can be accomplished.

The transmit host interface contains a data signal, write signal, FIFO count and a FIFO full indication. Correspondingly the receive interface contains a data signal, read signal, FIFO count and FIFO empty indications. This is shown in figure 1. This is a simple interface but still powerful enough to enable the implementation of most applications. Since the FIFO contents are transmitted and received in order without any additional processing a lot of freedom is left to the entity controlling the host interface.

The core also provides a time-code interface which basically transmits what is presented on the time input when the tickin input is asserted. For received time-codes tickout is asserted and the received time-code is presented on the time-output. No time-code validity checking is done (except for parity). This is left up to the user to implement externally if needed.

Separate control signals are also provided for the frequency during initialization and run-state. The former also controls the time-out periods used in the link interface FSM.

As for all other cores in the GRLIB IP library the core is written in an technology independent way with wrappers for the instantiation of technology dependent modules if needed by the target technology.

The presented features in total gives a fully compliant codec implementation. The following sections will show new features which make the core more versatile.

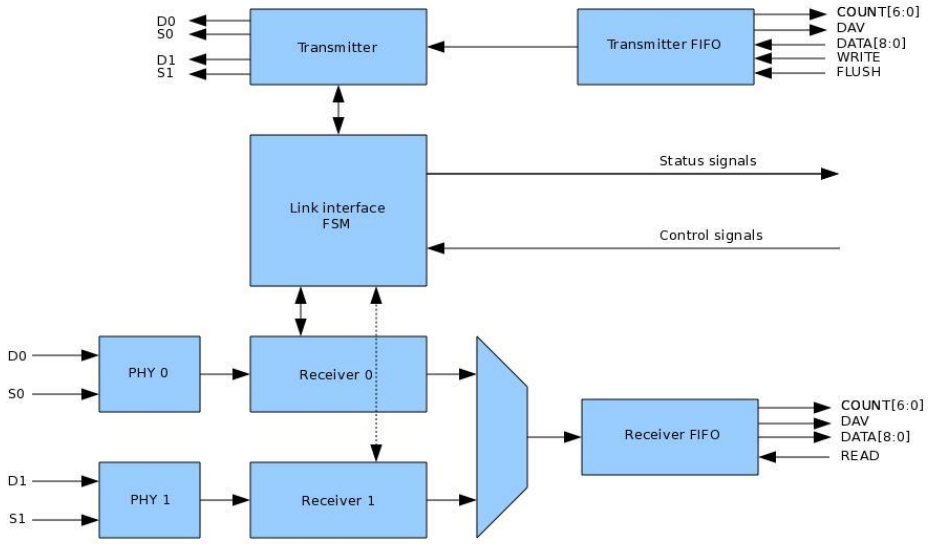


Figure 1: SpaceWire Codec block diagram

EXTERNAL SPACEWIRE INTERFACE

The signal interface to the SpaceWire network consists of a set of data and data-valid signals. These signals should be connected to a GRSPW2_PHY IP core also provided by Aeroflex Gaisler which handles the actual low-level clock and signal recovery or connecting to an external PHY component. This core is also used by the GRSPW2.

The benefit of having this in a separate core is that the data is always transferred in the same manner at the core interface not requiring any changes in the main core if the low-level implementation needs to be changed.

Another important issue to consider is the application of constraints during synthesis and place and route. Names of instances often change between versions of tools and the deeper in the hierarchy the instance is the more often the constraints need to be changed. By keeping the parts with a high implementation complexity in a single entity at the top-level of the design makes it easier to apply constraints.

On the receiver side the SpaceWire codec supports the standard self-clocked version with clock-recovery using an XOR gate. In addition to this there are three alternatives: single data rate (SDR) sampling, double data rate (DDR) sampling and an interface to the Aeroflex SpW transceiver [6]. The latter is a standard part provided by Aeroflex Colorado Springs which enables the codec to run at a reduced clock frequency compared to the SpaceWire link.

The sampling versions sample the incoming strobe and data signals and the sampling frequency must be at least 1.5 times higher than the maximum bit rate on the SpaceWire link. In DDR mode the clock frequency can be reduced compared to the

SDR version with the same sampling rate. The benefit of the sampling version is the much simpler constraint requirement. Only a single clock frequency constraint is needed to get a working implementation.

On the transmitter side SDR, DDR and Aeroflex SpW transceiver interfaces are supported. The DDR interface offers a reduced clock frequency compared to the SDR at the same bitrate but is not available for all technologies.

DUAL PORTS

The SpaceWire codec also provides an optional second SpaceWire port. The core can automatically switch between the ports depending on link activity or it can be forced using a specific link using a signal. The use of dual ports requires an additional GRSPW2_PHY IP core and internal receiver logic giving a small area penalty. For the Aeroflex SpW transceiver it would also require a second transceiver chip externally.

CONCLUSION

The SpaceWire codec presented is a technology independent and versatile core. It provides a fully standard compliant codec, and in addition to this several features such as dual ports and support for several different SpaceWire physical layers. The latter two features are not found in most other encoder-decoder implementations and make the core an area efficient alternative for redundancy applications, and easily movable between technologies.

REFERENCES

1. AMBA specification, Rev 2.0, ARM IHI 0011A, Issue A, ARM Limited, www.arm.com
2. GRLIB IP Core User Manual, Aeroflex Gaisler, www.gaisler.com
3. Remote Memory Access Protocol, ECSS-E-ST-50-52C
4. Marko Isomäki et al., “A Configurable SpaceWire Router VHDL IP Core”, ISC 2010
5. SpaceWire – Links, Nodes, Routers and Networks, ECSS-E-ST-50-12C
6. UT200SpWPHY01 SpaceWire Physical layer transceiver, Aeroflex Colorado Springs, www.aeroflex.com
7. SpaceWire - CCSDS packet transfer protocol, ECSS-E-ST-50-53C

A CONFIGURABLE SPACEWIRE ROUTER VHDL IP CORE

SpaceWire Components (Poster)

Short Paper

Marko Isomäki, Sandi Habinc

Aeroflex Gaisler, Kungsgatan 12, SE-411 19 Göteborg, Sweden

marko@gaisler.com, sandi@gaisler.com

ABSTRACT

Routers are an integral part of most SpaceWire networks and many are available as discrete components or IP cores from several providers. Aeroflex Gaisler has developed a highly configurable SpaceWire router VHDL IP core to meet the needs for technology independent router designs.

The main design goals have been configurability, technology independence, support of all standardized features and expandability.

This paper will give a short overview of the basic features and focus on how the design goals have been achieved and how the resulting implementation is useful.

INTRODUCTION

Currently there are only a few SpaceWire routers available either as discrete components with a single configuration or IP cores written in a hardware descriptive language (HDL). Some lack full support for one or more features such as packet distribution and group adaptive routing.

Aeroflex Gaisler provides a library of IP cores centered around the AMBA on-chip bus [1]. This library is designed in a technology independent manner with all technology dependent modules used through technology independent wrappers if needed by the target technology.

This paper presents the design of a SpaceWire router VHDL IP core designed for the GRLIB IP library to provide a router core which is technology independent, highly configurable and provides full standard support.

BASIC FEATURES

1.1 STANDARD SUPPORT AND ROUTING CAPABILITIES

All of the optional features mentioned in the SpaceWire standard [2] are supported to some extent. The core supports all the basic features with optional header deletion and individually assignable ports for each logical address. A port equivalence register is also provided for each physical or logical address which can be used to configure the router to be able to send a packet on the determined destination port or any port mark equivalent. In other words this is an implementation of group adaptive routing.

Interval labeling is a name given to a routing table with a consecutive ranges of logical addresses going to the same port thus enabling a simpler routing table as

claimed in the SpaceWire standard. The SpaceWire router supports ranges but this does however not result in a simplified table because the core supports the more general case with any address combinations being routable to the same port of which ranges is a special case.

Another optional feature which is not commonly found in current routers is packet distribution. The core contains a register for each physical or logical address which determines that an incoming packet with a certain destination address should be distributed to additional ports.

1.2 CONFIGURATION

Configuration is provided through port 0 as defined in the standard. RMAP [3] reads and writes are used to access the configuration which are located at static range of RMAP addresses. An experimental implementation of the draft SpaceWire plug and play protocol [4] can also be optionally enabled. It is also accessible through port 0 but is distinguished from RMAP through the protocol ID.

The core also has the option to use an AMBA APB interface to access the configuration registers which is more efficient than sending RMAP packets through the AMBA AHB port which would otherwise be the case.

1.3 SWITCH MATRIX

Connecting the ports together is a switch matrix where each port can be connected to any of the other ports using wormhole routing as required by the SpaceWire standard. However, many router designs have duplicated routing tables so that destinations for incoming packets can be determined immediately without contention. The downside of this is a large area overhead since the routing table has to be duplicated once for each port.

The Aeroflex Gaisler router instead has a single routing table with pipelined access to avoid timing problems. One new destination can be determined each clock cycle and should be sufficient without any significant performance penalty with a normal workload. The case where a performance penalty would occur is when many or all ports are simultaneously sending very short packets but this scenario should be unlikely to occur at any great frequency.

PORTS

1.4 SPACEWIRE PORTS

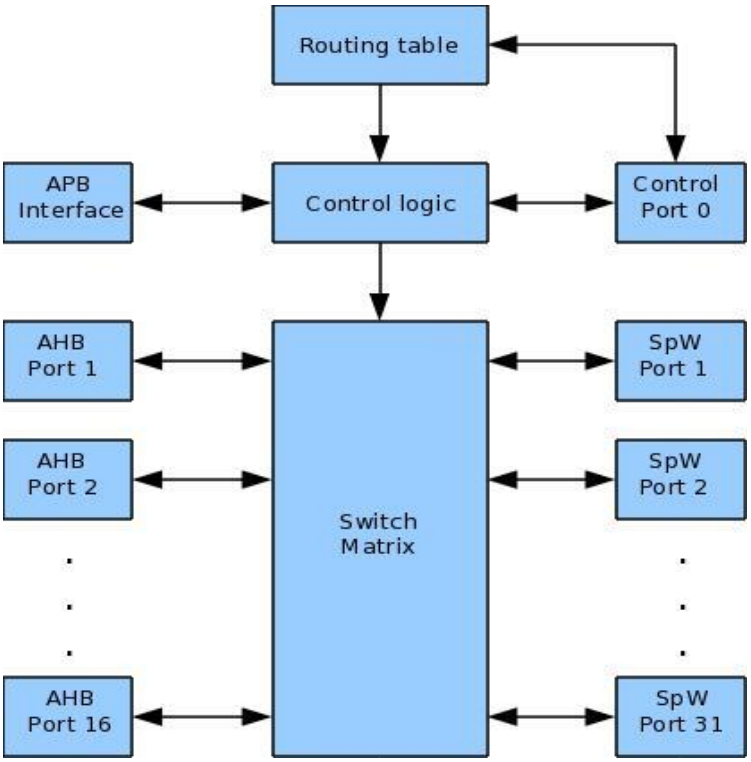


Figure 1: SpaceWire router block diagram

The total number of ports is configurable from 2 to 31. Up to 31 can be external SpaceWire ports where each port is an instantiation of the SpaceWire codec core [5] provided in GRLIB [6]. The SpaceWire codec used for the SpaceWire ports provides many different output and input physical layer implementations. Among them are self-clocking reception, sampling, SDR or DDR sampling inputs or outputs and an Aeroflex SpW transceiver interface [6]. The different interfaces each have their strong features which makes them suitable for different technologies and link speeds. This makes the core versatile in respect of technology independence.

1.5 AMBA AHB PORTS

Up to 16 of the ports can be DMA engines with optional RMAP targets which transfer packets on an AMBA bus of a SpaceWire network. The AMBA AHB interfaces are based on the GRSPW2 [6]. The number of ports cover the complete range of what the standard allows. Through the use of one or more AHB interfaces connected to the router ports the router can easily be expanded with existing GRLIB cores. For example MIL-1553, CAN, PCI or Ethernet cores can for example be connected to the DMA engine providing a bridge to one or more of these interfaces with a minor

amount of design work. In the same way the external parallel interface that is common in router components can be implemented.

CONCLUSION

The Aeroflex Gaisler SpaceWire router is a highly configurable and technology independent core. It supports all of the features mentioned in the SpaceWire standard with some additional features. Due to the technology independent design and reuse of existing IP it is easy to move the design between technologies and to reconfigure a design. Due to its connection with the GRLIB IP library it can also be expanded with bridges to a wide variety of on-board buses.

REFERENCES

1. AMBA specification, Rev 2.0, ARM IHI 0011A, Issue A, ARM Limited, www.arm.com

2. SpaceWire – Links, Nodes, Routers and Networks, ECSS-E-ST-50-12C

3. Remote Memory Access Protocol, ECSS-E-ST-50-52C

4. SpW PnP Protocol definition, Draft A v2.1, Space Technology Centre, University of Dundee, http://spacewire.esa.int

5. Marko Isomäki et al., "A versatile SpaceWire Codec VHDL IP Core", International SpaceWire Conference, St Petersburg, 2010

6. GRLIB IP Core User Manual, Aeroflex Gaisler, www.gaisler.com

THE EVOLUTION OF SPACEWIRE: A COMPARISON TO ESTABLISHED AND EMERGING TECHNOLOGIES

Session: SpaceWire Networks and Protocols (Poster)

Short Paper

Robert A. Klar, Allison R. Bertrand

Southwest Research Institute, San Antonio, TX, 78238

E-mail: robert.klar@swri.org, allison.bertrand@swri.org

ABSTRACT

Designed for use in space applications, SpaceWire offers many advantages over other comparable communications technologies. It requires relatively simple circuitry to implement, offers low power consumption, and supports high link speeds. It has rapidly gained acceptance and has been successfully employed in support of a wide variety of missions.

Since SpaceWire was standardized in 2003, it has been supplemented by several higher-level protocols. Standards were recently published which specify two such protocols: the Remote Memory Access Protocol (RMAP) and the Consultative Committee for Space Data Systems (CCSDS) Packet Transfer Protocol (PTP). With anticipated introduction of a standard for SpaceWire-RT (SpaceWire Reliable Timely), mechanisms are introduced for SpaceWire to provide for timely delivery of information with Quality of Service (QoS) guarantees. A new standard, known as SpaceFibre, which is derived from SpaceWire and supports even higher transfer speeds is currently in development.

This paper explores the evolution of SpaceWire over time. It compares and contrasts SpaceWire with other high-speed technologies such as Asynchronous Transfer Mode (ATM), Switched Ethernet, Peripheral Component Interconnect (PCI) and Rapid I/O and discusses the trade-off between design complexity and performance.

1 INTRODUCTION

SpaceWire is a simple, low-power, high-speed networking technology derived originally from the IEEE 1355 standard [1]. It has rapidly gained widespread acceptance for use in onboard spacecraft communication systems because it provides many advantages and is easy to implement in radiation-tolerant programmable logic devices. The SpaceWire standard replaced the IEEE 1355 physical layer with low-voltage differential signaling (LVDS) which is more suitable for the harsh space environment [2]. It uses point-to-point links to connect nodes rather than a shared bus and provides much flexibility for incorporating redundancy. SpaceWire has relatively low memory requirements because it uses a wormhole routing technique in which a packet received on an input of a switch begins retransmission on an output before it is completely received.

Since SpaceWire was first published as a standard by the European Cooperation for Space Standardization (ECSS) in January 2003 [3], it has evolved and several new standards based on SpaceWire have emerged. In July 2008, the SpaceWire standard was republished but no changes were made to the technical content [4]. In February 2010, ECSS published a set of supplementary standards including "Space engineering: SpaceWire protocol identification" [5], "Space engineering: SpaceWire – Remote memory access protocol" [6], and "Space engineering: SpaceWire – CCSDS packet transfer protocol" [7]. In addition to these published standards, a number of proposed draft standards are under development. These draft standards include SpaceWire-RT, which extends SpaceWire to include Quality of Service (QoS) guarantees [9], and SpaceWire-PnP (Plug-and-Play), which provides support for device discovery, network management, and device configuration services. As SpaceWire has evolved, it has developed capabilities similar to many new and established network protocols.

2 COMPARISONS TO OTHER PROTOCOLS

2.1 ASYNCHRONOUS TRANSFER MODE

Asynchronous Transfer Mode (ATM) is a networking technology originally designed for efficient transfer of voice, video, and data that was standardized by the International Telecommunication Union Telecommunication Standardization Sector (ITU-T) [8]. ATM uses a connection-oriented model in which a virtual circuit is established between endpoints. In order to support QoS guarantees, ATM switches fixed-size transfer units, called cells, which provide constant transmission delays and guaranteed capacity.

In contrast to the fixed-size cell used by ATM, SpaceWire uses variable-size transfer units, called packets. The SpaceWire standard does not restrict the maximum size of a transfer unit (an approach inherited from IEEE 1355 [1]). Consequently, long packets can cause delay problems in a SpaceWire network as they can block links in their path. Although this can be a serious issue, the size of transfer units is often limited in practice by higher-level protocols.

SpaceWire-RT (SpaceWire Reliable Timely) has been proposed as a higher-layer protocol for SpaceWire to provide a suitable solution where QoS guarantees are needed [9]. SpaceWire-RT supports a scheduled system by segmenting SpaceWire packets into maximum size transfer units, called DPs. DPs are scheduled using fixed-time slots, similar to the multiplexing of cells in ATM. SpaceWire-RT, also like ATM, requires resource reservation in order to establish a timely path through the network.

2.2 ETHERNET

Ethernet is the ubiquitous, asynchronous, packet-switched network technology which dominates the Internet today. Ethernet uses variable-size packets called frames. Similar to SpaceWire-RT, Ethernet restricts transfer units to a maximum size. Early Ethernet implementations used a scheme known as carrier sense multiple access with collision detection (CSMA/CD) to share a common communications channel. Under this scheme, if two senders transmit simultaneously causing a collision, both senders would terminate transmission, delay for a brief random time and then attempt to

transmit again. Although this limits throughput, it is simple to implement. Most modern Ethernet implementations are switched. With this approach, Ethernet switches temporarily buffer frames and delay transmission in order to avoid collisions.

Perhaps one reason for Ethernet's immense popularity is that it integrates well with the Internet Protocol (IP). Ethernet conveniently provides support for a link-layer broadcast mechanism and includes network unique source/destination addresses in each frame. This greatly simplifies configuration by providing a mapping to directly support an Address Resolution Protocol (ARP).

SpaceWire too can be characterized as an asynchronous, packet-switched network. However, in contrast to Ethernet, SpaceWire does not provide provisions for a link-layer broadcast mechanism and does not inherently provide network unique addresses within a packet. Fortunately, these features can be added using higher-level protocols [10]. In this way, SpaceWire can effectively be integrated with the Internet Protocol.

2.3 Peripheral Component Interconnect

Peripheral Component Interconnect (PCI) refers to a series of related standards. PCI was originally introduced by the Intel Corporation for use in attaching hardware devices in a computer system. An industry organization, known as the PCI Special Interest Group (PCI-SIG), was formed in 1992 and released the PCI v2.0 standard in April 1993 [11]. The standard defined a 5-Volt, 33-MHz, 32-bit parallel architecture with throughput of 133 MB/s (>1Gbps).

The latest standard in the PCI family is PCI Express. Unlike previous PCI standards which were based on a shared parallel bus, PCI Express shifted to a serial point-to-point architecture in order to avoid issues with clock skew at high clock rates. PCI Express has started to displace conventional PCI in modern day Personal Computers (PCs).

In the PCI family, PCI Express is the most similar in architecture to SpaceWire. PCI Express is typically implemented on a backplane. Backplane implementations of SpaceWire have been proposed and research has been conducted to evaluate connectors suitable for the harsh environments of space [12].

2.4 RapidIO

RapidIO is a high-speed, packet-switched interconnect technology that was designed for use in integrating components on a circuit board. The technology was originally developed as a collaborative partnership between Motorola and Mercury Computer Systems, Inc. with a 1.0 specification released in late 1999 [13]. In 2000, the RapidIO Trade Association was formed to direct development and encourage adoption of the technology. RapidIO exists in Serial and Parallel versions. Although Serial RapidIO achieves higher speeds than SpaceWire, it has not been qualified for use in space applications.

A new standard, known as SpaceFibre, which is derived from SpaceWire and supports even higher transfer speeds is currently in development [14]. SpaceFibre shares some common characteristics with Serial RapidIO such as the use of 8B/10B encoding to

transmit the clock along with the data signal. SpaceFibre requirements include the target or reaching link speeds of up to 2.5 Mbps which is comparable to RapidIO.

3 SUMMARY AND CONCLUSIONS

Table 1. Summary of Network Protocols

	SpaceWire	ATM	Ethernet	PCI	RapidIO
Standardized	2003	1988	1982	1993 (v2.0)	2001 (1.1)
Standards Body	ECSS	ITU-T	IEEE	PCI-SIG	RapidIO TA
Created By	ESA/ESTEC	ITU, ATM Forum, et. al.	Xerox	Intel	Motorola and Mercury Computer
Related Standards	RMAP, PTP, SpaceWire-RT, SpaceWire-PnP, SpaceFibre	SONET/SDH	10/100/1000BASET, 1Gb and 10Gb varieties	PCI v2.1, v2.2, v2.3, PMC, PCI-X, PCIe	RapidIO 1.1, 1.2, 1.3, 2.0, 2.1
Capacity	2 – 400 Mbps	155Mbps (OC-3), 10Gbps (OC-192)	10Mbps, 100Mbps 1Gbps, 10Gbps	1Gbps, 2Gbps, 4Gbps	1Gbps (x1) – 10Gbps (x4)
Type	serial point-to-point	serial point-to-point	shared-bus, serial point-to-point	parallel, 32 or 64 bits	serial point-to-point
Transfer Unit	packet	cell	frame	bus transactions	packet

SpaceWire has proven to be a simple, yet versatile networking technology. The creation of integrated higher-level protocols for SpaceWire has made it viable for a variety of practical applications.

4 REFERENCES

1. IEEE Std 1355-1995. "IEEE Std 1355-1995 Standard for Heterogeneous InterConnect (HIC) (Low Cost Low Latency Scalable Serial Interconnect)", Institute of Electrical & Electronics Engineers, Inc., June 1996.
2. ANSI/TIA/EIA-644-A. "Electrical Characteristics of Low Voltage Differential Signaling (LVDS) Interface Circuits", Telecommunications Industry Association, February 2001.
3. ECSS-E-50-12A, "Space Engineering: SpaceWire – Links, nodes, routers, and networks", ESA-ESTEC, January 2003.
4. ECSS-E-ST-50-12C, "Space Engineering: SpaceWire – Links, nodes, routers, and networks", ESA-ESTEC, July 2008.
5. ECSS-E-ST-50-51C, :Space Engineering: Space engineering: SpaceWire protocol identification", ESA-ESTEC, February 2010.
6. ECSS-E-ST-50-52C, "Space engineering: SpaceWire – Remote memory access protocol", ESA-ESTEC, February 2010.
7. ECSS-E-ST-50-53C, "Space engineering: SpaceWire – CCSDS packet transfer protocol", ESA-ESTEC, February 2010.

8. ITU-T I.361. "ATM Layer Specification", International Telecommunication Union Telecommunication Standardization Sector, February 1999.
9. Steve Parkes and Albert Ferrer, "SpaceWire-RT", International SpaceWire Conference 2008, Nara, Japan, November 2008.
10. Robert Klar, et. al. "Integration of Internet Protocols with SpaceWire using an Efficient Network Broadcast", International SpaceWire Conference 2007, Dundee, Scotland, November 2007.
11. http://www.pcisig.com/specifications/PCI_Family_History.pdf, PCI-SIG Developers Conference, 2006.
12. Koji Sibuya, et. al. "Evaluation And Analysis Of Connector Performance For The Spacewire Back Plane", International SpaceWire Conference 2008, Nara, Japan, November 2008.
13. http://www.rapidio.org/about/backgrounder. Rapid I/O Trade Association. 2010.
14. Steve Parkes, et. al. "SpaceFibre", International SpaceWire Conference 2008, Nara, Japan, November 2008.

LVDS IP-BLOCKS FOR HIGH SPEED DATA TRANSMISSION IN SPACEWIRE SYSTEMS

Session: SpaceWire Components (Poster)

Short Paper

Sergey Kondratenko, Valery Baikov, Yury Gerasimov, / *National Research Nuclear University "MEPhI" 31, Kashirskoe sh., Moscow 115409, Russia*

Tatiana Solokhina / *ELVEES R&D Center of Microelectronics, stroenie 2, proezd 4922, Yuzhnaya promzona, Zelenograd, Moscow124460, Russia*

E-mail: kondsv@kaf3.mephi.ru, baikov@kaf3.mephi.ru, germ@kaf3.mephi.ru, tanya@elvees.com

ABSTRACT

The report deals with the special features and results of designing LVDS transmitters – receivers, manufactured by the CMOS 0.25/0.18/0.13 μm technologies for series data transmission at a speed up to 1 Gbit/s. The prospects of raising the transmission speed up to 1.5-2 Gbit/s are considered.

1 INTRODUCTION

LVDS (low-voltage differential signaling) – is a topical format of data presentation in high-speed communication interfaces, implemented in the Space Wire standard [1]. The peculiarity of the standard consists in the communication line's low voltage differential signal, being a high frequency bit flow. The problem of raising the LVDS transmission speed, while retaining a relatively low power consumption, is one of the most topical ones in practice [2,3].

The initial requirements predetermine the presence of a high-voltage periphery in the body of the transmitter or receiver – some parts of CMOS circuitry with the 0.35 μm design rules and 3.3 V supply voltage. The basic circuit elements are fed by standard supply voltages of 2.5/1.8/1.3 V (in accordance with the 0.25/0.18/0.13 μm technologies).

High-voltage parts (0.35 μm) constitute a considerable fraction of the whole transmitter/receiver design and essentially influence the speed characteristics of interface. For the sake of unification they are implemented in the form of structure-layout cores and are compatible with each of the chosen technologies 0.25/0.18/0.13 μm.

High-speed voltage translators with small signal delay have been designed for interfaces with high-voltage elements.

The Space Wire standard determines the high rate of switching the differential signal levels – of exchanging the «0»-s и «1»-s in the sequence of bits transmitted. Nevertheless, the developed circuits can function at slow bit rates also (for instance, at 10 Mbit/s) – down to the transmission of a constant differential signal «0» or «1».

An additional (service) function of the LVDS receiver is the detection of breakage in the communication line with the LVDS transmitter.

All the receivers/transmitters design can switch into the mode of reduced power consumption.

2 LVDS TRANSMITTERS

The transmitter converts the input series of bits, presented by the “0” and “1” logical levels, into the differential output LVDS signal.

The initial data and control signal level (permission of operation, transfer to the mode of reduced power consumption) is carried out by logical elements, being fraction of the low-voltage part of the circuit. The conversion into differential form and translation of voltage logic levels, necessary thereby, are carried out by high-voltage analog elements.

Fig. 1 shows the principle of shaping the Up, Un differential signal in communication line. For simplicity the shaping circuit elements are presented as macromodels.

The transmitters developed are intended for communication systems with transmission speeds up to 600 Mbit/s. This parameter is maintained for capacitive loads up to 5 pF at each of the two line busses Up, Un. Reducing the capacitance down to 1.5 pF permits to raise the transmission rate up to 2 Gbit/s (fig. 2).

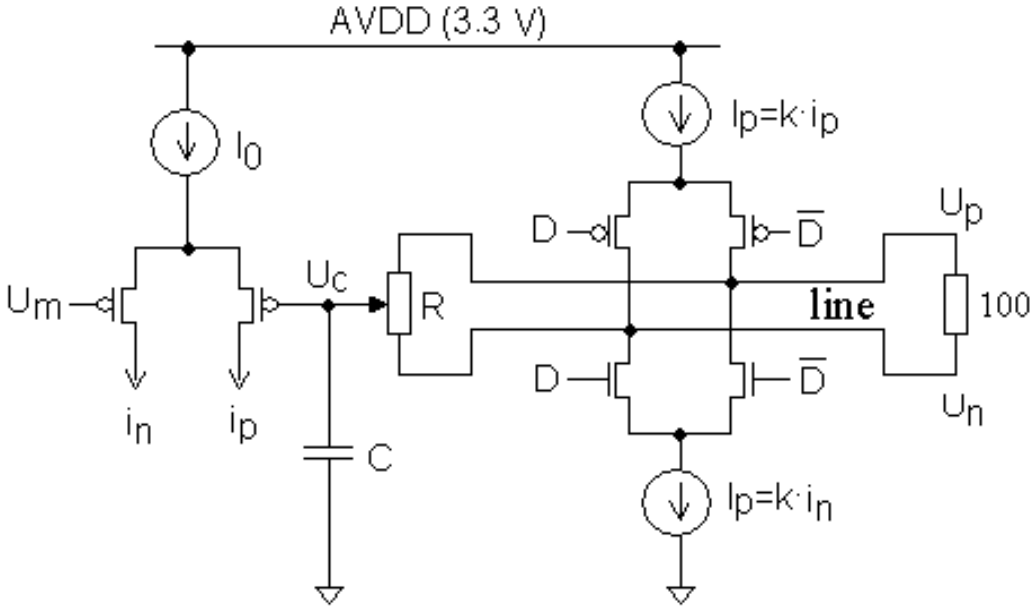


Fig. 1. Circuit, shaping the LVDS differential signal in transmitter

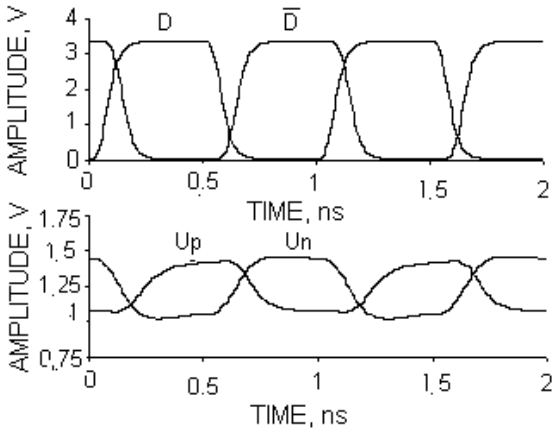


Fig. 2. Time diagrams of transmitter (of the circuit, shaping the LVDS signal) at a speed of 2 Gbit/s

3 LVDS RECEIVERS

Such a receiver converts the differential LVDS signal (Up, Un) in a digital series of bits (OUT).

The basic element configuration of the receiver circuit (macro) is shown in fig. 3. Time diagram of receiver is shown in fig. 4.

An effective way of extending the high-frequency band of receiver, bound with shaping the Uinv(t) voltage, consists in using a ring structure of current mirrors in the basic circuit of fig. 3 (that was studied at simulation). The switching process acquires threat a regenerative character, what allows us to determine distinctly the temporal limits of bits. The result becomes obvious, when comparing fig. 4b to fig. 4a (at similar characteristics of the signal transmitted). The use of a ring of current mirrors eliminates the necessity of an automatic adjustment circuit, while the rate of data reception may be raised to 2.5-3 Gbit/s.

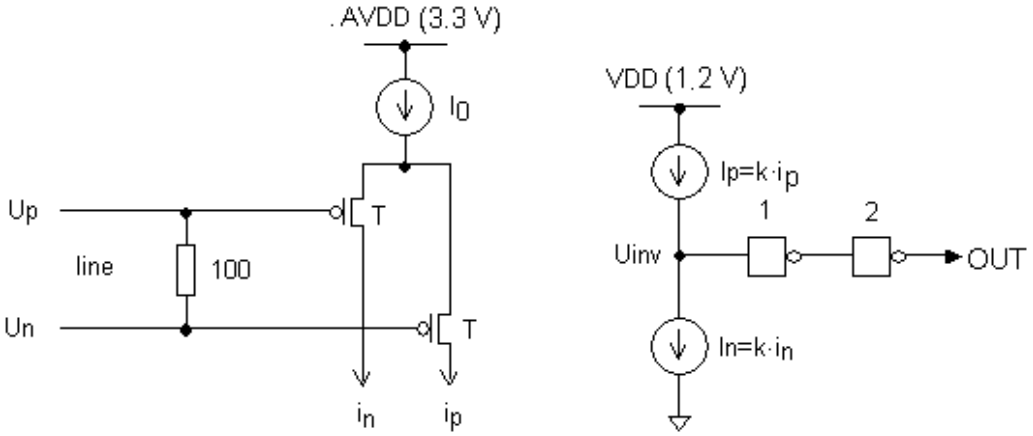
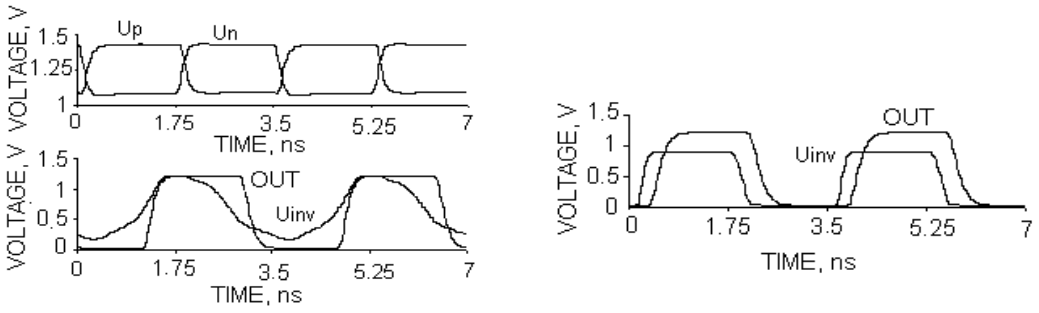


Fig. 3. Circuit diagram of converting the differential LVDS signal into a digital one



a) b)

Fig. 4. Time diagrams of receiver at an initial structure (a)
and at a ring structure of current mirrors (b)

4 EXPERIMENTAL RESULTS

Three versions of the LVDS transmitters-receivers have been manufactured with CMOS – 0.25/0.18/0.13 μm technologies. The experimental studies confirmed their characteristics, as complying with the simulation results and requirements of standards (including the value of current consumption in the modes of operation and of reduced power consumption) for transmission speeds up to 1 Gbit/s.

5 CONCLUSION

The peculiarities of the LVDS interface have been considered, as a building block of the SpaceWire standard.

The features of designing LVDS transmitters-receivers have been described in detail. The basic attention was paid to provide a high speed of those units, allowing to achieve transmission speeds of 1 Gbit/s and higher, which redundantly satisfy the needs of SpaceWire applications. In the course of designing the LVDS transmitters-receivers there was applied an original technique, invariant relative to the specific version of the submicron CMOS-technology used (with rules 0.25/0.18/0.13 μm).

6 REFERENCES

1. Telecommunications Industry Association, “Electrical Characteristics of Low Voltage Differential Signaling (LVDS) Interface Circuits”, ANSI/TIA/EIA-644-1995, March 1996.
2. Oksiucik M., Gerka S., Baranowicz K., Kowalewski M., Sobala B., Lewandowski M., Matczak G. “A CMOS 0.13 Micrometer Realization Of External Component Free 1Gbps LVDS Driver”, Proceedings of the International Conference Mixed Design of Integrated Circuits and System, MIXDES 2006, p.188-191.
3. Silva-Martinez M. C., Nix J., Robinson M. “Low-voltage low-power LVDS drivers”, IEEE Journal of Solid-State Circuits, v.40, Feb. 2005, p.472-479.

VERIFICATION OF HIGH RESOLUTION TIMING SYSTEM WITH SPACEWIRE NETWORK ONBOARD ASTRO-H

Session: Missions and Applications (Poster)

Short Paper

Tomomi Kouzu, Yukikatsu Terada, Kaori Iwase, Makoto S. Tashiro,

Department of Physics, Science, Saitama University, Shimo-Okubo 255, Sakura-ku, Saitama 338-8570, Japan

Takayuki Yuasa,

Department of Physics, School of Science, The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-003, Japan

Masaharu Nomachi,

Department of Physics, Graduate School of Science, Osaka University, 1-1 Machikaneyama, Toyonaka, Osaka 560-0043, Japan

Yoshitaka Ishisaki,

Department of Physics, Tokyo Metropolitan University, 1-1 Minami-Osawa, Hachioji, Tokyo 192-0397, Japan

Tadayuki Takahashi, Motohide Kokubun, Masanobu Ozaki,

Institute of Space and Astronautical Science, Japan Aerospace Exploration Agency (ISAS/JAXA), 3-1-1 Yoshinodai, Sagamihara, Kanagawa 229-8510, Japan

and ASTRO-H collaborations

E-mail: kouzu@heal.phy.saitama-u.ac.jp, terada@phy.saitama-u.ac.jp, iwase@heal.phy.saitama-u.ac.jp, tashiro@phy.saitama-u.ac.jp, yuasa@juno.phys.s.u-tokyo.ac.jp, nomachi@lns.sci.osaka-u.ac.jp, ishisaki@phys.metro-u.ac.jp, takahasi@astro.isas.jaxa.jp, kokubun@astro.isas.jaxa.jp, ozaki@astro.isas.jaxa.jp

ABSTRACT

ASTRO-H is an X-ray observatory satellite exploring in the universe to be launched in 2014. The data acquisition system of this mission will be constructed based on the SpaceWire network. One of the most important issues is to evaluate the time assignment system on the SpaceWire network. In this paper, detail descriptions of timing system of ASTRO-H are shown, as well as verification experiments performed with SpaceWire modules. Based on the obtained results, the timing accuracy of the ASTRO-H system is estimated to be a few microseconds, which is well below the requirement.

1 INTRODUCTION

1.1 THE ASTRO-H MISSION

ASTRO-H [1] is the 6th satellite in Japanese X-ray observatory series to be launched in 2014. It is a Japan-US mission leaded by JAXA and NASA, and collaborated with

ESA. The data acquisition system of ASTRO-H is required to have high timing capability with timing resolution of about 10 μs and accuracy of about 30 μs in order to archive the scientific goals of the mission, for example, high energy physics from observations of astrophysical objects showing rapid variability like pulsars, black holes etc.

1.2 Data Acquisition System of the SpaceWire Network in ASTRO-H

The command and telemetry communication onboard ASTRO-H will be constructed on the SpaceWire network. Figure 1 shows the current design of the network onboard ASTRO-H. The boxes in the figure except for that denote as GPSR (GPS Reciever) represent the network nodes having SpaceWire interfaces. The messages transmitted between nodes, "Space Packets", are sent via the RMAP protocol. To guarantee the quality of service (QoS) of the data transfer, communication time is divided into 64 time slots by highly prioritized 64 Hz time-codes.

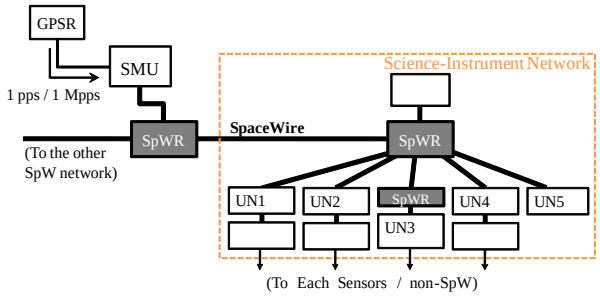


Figure- 1 : SpaceWire-based data acquisition system onboard ASTRO-H. GPSR, SpWR and UNs mean a GPS Reciever, SpaceWire routers and User Nodes (see text), respectively.

2 Design of the Timing System in the ASTRO-H Network

2.1 Distribution of the Spacecraft Time

Time when the space packet is generated on the spacecraft is to be assigned as a UTC (Universal Time, Calculated) value on ground. Onboard the spacecraft, the data-acquisition system delivers a monotonously increasing time counter called "TI". A node performed as a “Time Master” [2] distributes TIs to all the nodes, and all the components onboard share the single time source, TI. On the ground station, UTC from caesium clocks and the satellite TI monitored via real time telemetry are compared to produce a conversion table. This method is previously used in scientific satellites of JAXA, like the *Suzaku* satellite [3].

In the ASTRO-H network, the top node called Satellite Management Unit (SMU) has the function of “Time Master”, where TIs are generated from 1 pps and 1 Mpps clocks of GPSR. Then, the Time Master distributes the TI values to the other nodes. We call these slave nodes as “User Nodes”. In order to distribute TIs properly, the TI is divided into two parts: one is 32-bit “TIME DATA” above a second, which is distrubuted via “RMAP write”, and the other is 6-bit Time-Code for sub-second distributed by the time-code manner [2]. With this method, the accurate timing of the leading edge of the time clock of TI is notified by the Time-Code, and thus, the jitters in distributing the Time-Code [4] will limit the timing accuracy of this system.

2.2 TIME ASSIGNMENT OF X-RAY EVENTS

ASTRO-H carries X-ray sensors as the User Nodes. They detect X-ray photons coming from celestial objects randomly. The arrival time is required to be assigned event by event with the timing accuracy of 10 μs. Therefore we need finer time counters at the User Nodes for the time assignments of X-ray events. We call them "LocalTime". Although LocalTime can be simply realized by a phase locked loop to the TI counter, we implement LocalTime with free-run clocks to reduce required resource in each User Node. The information of each X-ray event carries LocalTime value, and such event data are gathered into a space packet with a corresponding TI value. To assign time of each event, we should know a mapping table between TI and LocalTime. This mapping table is acquired periodically by a constant time interval (hereafter, Sample Period), which is summarized into the House-Keeping space packet. In order to realize this system with a sufficient accuracy, we should determine the Sample Period referring the stability of the quartz oscillators onboard the spacecraft under the expected temperature circumstances.

3 VERIFICATION OF THE TIMING SYSTEM

3.1 AIMS OF THE VERIFICATION EXPERIMENTS

The aim of the experiment is to examine the concept of the time assignment system described in section 2. As already mentioned in section 2.2 and 2.3, we should check the following two items with the actual hardware: (i) jitters of Time-Codes and (ii) stability of LocalTimes.

3.2 SET UP AND ANALYSIS

The configuration of the experiment is shown in figures 2 and 3. The link rate between Time Master and User Node was set to be 100 MHz without any other data transfer; i.e, it was in communication of NULL (8 bits). In experiment (i), time lags between Time-Code “ticks” of Time Master and User Node were taken by a Time-to-Analogue Converter with and without the router hop. In experiment (ii), the mapping table of TI and LocalTime was recorded on the User Node into its own SDRAM every second, and was read out via a CPU node named SpaceCube [5]. The User Node was set into a thermostat bath under the temperature of 10 ±5, ±10, or ±20 ºC with no router between the two nodes. Then, we calculated the interpolation function from data sampled every Sample Period, and we got the residuals (hereafter “Timing errors”) by comparison with the function and the original fine data.

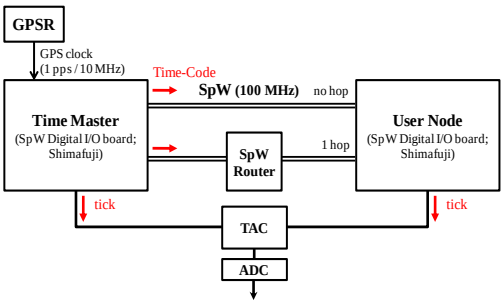


Figure- 2 : Set up of the experiment (i).

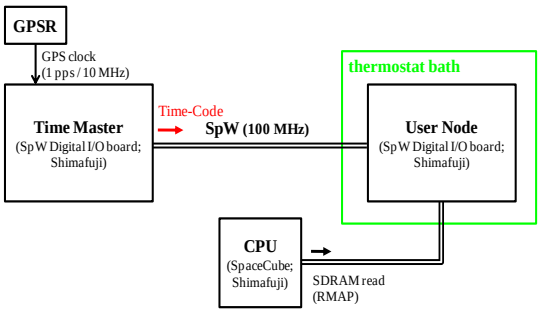


Figure- 3: Set up of the experiment (ii).

3.3 RESULTS AND DISCUSSIONS

Figure 4 shows the results of the experiment (i); histograms of time delays of Time-Codes between Time Master and User Node. Full widths of the distributions are 90 and 160 ns without and with the router between the two nodes, respectively. These values can be interpreted as time jitters of 70 or 140 ns by one or two routers, respectively. The rest 20 ns is due to other unknown origin. In other words, N hops causes 70 × N ns jitter. In the ASTRO-H configuration, the minimum link rate is to be 20 MHz so that jitters of Time-Codes should be 70 ns × (100 MHz / 20 MHz) = 350 ns at 1 hop. In the ASTRO-H network, 4 or 5 hops are expected (figure 1), and thus the total jitters of Time-Codes are expected to be 1.4-1.8 μs. This value is well below the timing accuracy required.

Figure 5 shows the results of the experiment (ii); Timing errors as a function of Sample Periods. Even if the temperature varies ±20 ºC, timing distortion by sampling the mapping table is less than the requirement.

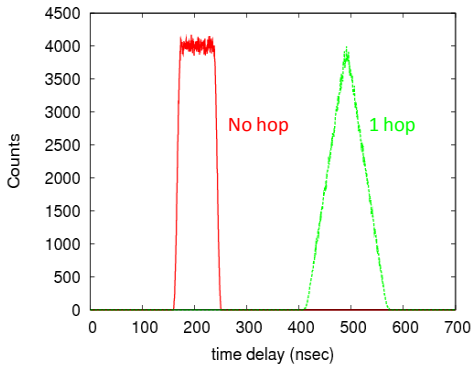


Figure- 4: Histograms of jitters of Time-Codes.

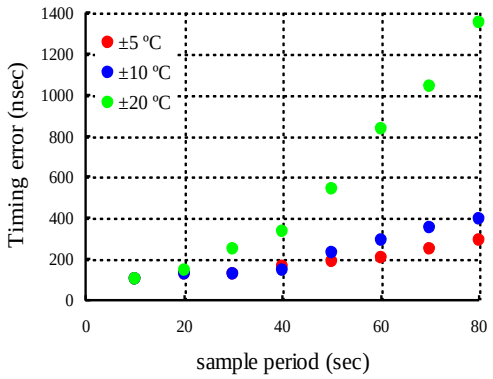


Figure- 5: Timing errors an a function of Sample period, under temperature variety of ±5, 10, 20 K (red, blue and green, respectively).

In summary, the method proposed by us fulfils required time performance to the ASTRO-H satellite.

4 REFERENCES

[1] T. Takahashi et al., “The NeXT mission”, Proc. SPIE, 7011, 70110-O, 2008

[2] S. M. Parkes, “The Operation and Uses of the SpaceWire Time-Code”, International SpaceWire Seminar, 4-5 November 2003, ESTEC Noordwijk, The Netherlands

[3] Y. Terada et al., “In-Orbit Timing Calibration of the Hard X-Ray Detector on Board Suzaku”, Publications of the Astronomical Society of Japan, 60, 25

[4] C. Christophe & P. Frederic, “SpaceWire in the SIMBOL-X Hard X-ray Mission”, International SpaceWire Conference 2008, 4-6 November 2008, Nara, Japan

[5] T. Yuasa et al., “A Portable SpaceWire/RMAP Class Library for Scientific Detector Read Out Systems”, International SpaceWire Conference 2008, 4-6 November 2008, Nara, Japan

A FORMAL METHOD FOR VERIFYING THE IMPLEMENTATION OF SPW DATA-STROBE-ENCODING BY APPLYING THEOREM PROVING

Session: SpaceWire Test and Verification (Poster)

Long Paper

Liming Li

College of Information Engineering, Capital Normal University,

Beijing, China, 100048

Liya Liu

Dept. of Electrical and Computer Engineering, Concordia University,

Montreal, Quebec, Canada, H3H 1M8

Yong Guan, Yan Zhang

College of Information Engineering, Capital Normal University,

Beijing, China, 100048

Jie Zhang

College of Information Science & Technology, Beijing University of Chemical Technology, Beijing, China, 100029

Limin Tao

Beijing Engineering Research Center of High Reliable Embedded System

Beijing, China, 100048

E-mail: liliminga@126.vom, liy_liu@ece.concordia.ca,guanyxxxy@263.net, anythingroom@126.com, jzhang@mail.buct.edu.cn

ABSTRACT

For the requirement of building a highly reliable communication system, SpaceWire was applied in Space Solar Telescope (SST) project completed by National Astronomical Observatories, Chinese Academic of Sciences. This paper is based on part of work on SST project. In SpaceWire standard, Data-Strobe (DS) encoding was an encoding scheme for transmitting data in digital circuits. This study aimed to verify that the DS encoder circuit, which was developed for SST project, faithfully implemented the specification in SpaceWire standard. Equivalence checking was applied between the specification and implementation. With the aid of HOL tool, this

equivalence checking was carried out in a formal verification method, theorem proving. According to the requirements of the standard, a primitive recursive function was defined using Meta Language (ML) in order to specify the circuit. Then the components implemented in VHDL code were modeled with predicates. The result showed that the implementation is equivalent to the specification. It suggested that this DS encoder circuit implemented on FPGA can be applied reliably in the SST project.

1 INTRODUCTION

With the development of technology and science, the density of the circuits increases quickly. The extensive applications of the circuits are found in transportation systems, medical applications, defense systems, and so on, nearly all aspects. It is well known that the cost of a failure is unacceptably high [3], especially in aerospace safety domain. Any subtle error in the design might cause unexpected trouble under a particular set of conditions. It may mean the loss of life or serious impairment for many human beings, the disastrous consequences. Thus, to verify the correctness of the circuit becomes imperative.

Generally, designers try to ensure correctness through simulation and testing. However, it is impossible to test or simulate all the cases for large, complex systems. Furthermore, simulation and testing inputs are usually designed to detect only certain well-defined types of faults. Some corner cases might be ignored during the process of design, simulation and testing. And exhaustive simulation and testing are no longer possible because of the increasing complexity of the circuit. [3]

Formal methods have the capability of conducting precise system analysis and overcome the limitation described above. It is a technology to construct the system based on mathematical model formally. Theorem proving is one of the most commonly used formal methods. It allows to mathematically reason about system properties by representing the behavior of a system in higher-order logic in a general model. In this way, the specification and implementation are expressed as the general mathematical model so that all the cases are covered when they are proved to be equivalent.

In SST project, SpaceWire link addresses the handling of payload data and control information on boarding a spacecraft. DS encoder is the critical circuit part in SpaceWire standard for high speed data link. To verify that the design of this circuit is implemented reliably, theorem proving is applied. In chapter 2, the method is described in details. Then result is shown in chapter 3. Finally, it comes to the conclusion and presents the future work.

2 METHOD

For the purpose of verification, both the specification based on SpaceWire standard and the implementation based on the hardware are modeled abstractly. These two

models are called DS Encoding Specification and DS Encoding Implementation separately. Both of the models are defined formally utilizing ML (Meta Language) in higher-order-logic. The latter comes from the VHDL code and the gates are obtained based on their Boolean functionality. By applying the rules for reasoning in higher-order-logic, equivalence checking was applied between these two models.

2.1 DS ENCODING SPECIFICATION

In SpaceWire standard, Data-Strobe (DS) encoding is specified as the coding scheme which encodes the transmission clock with the data into Data and Strobe so that the clock can be recovered simply by implementing an exclusive OR operation on the Data and Strobe signals. It is illustrated in Figure 1. As it shows, Data are transmitted directly and the state of the Strobe signal changes whenever the data alters one data bit interval to the next. [1]

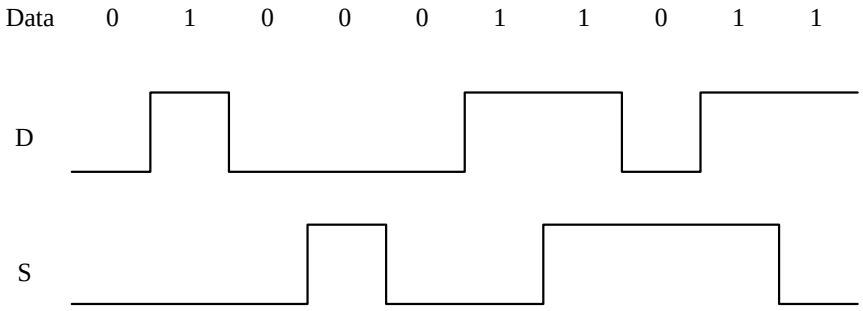


Figure 1 Data-Strobe (DS) Encoding

The major feature of DS encoding is that the Data are transmitted directly and the Strobe signal shall vary state whenever the Data does not change from one bit to the next. In order to demonstrate the signals in a general expression, in following definition, t denotes clock, and X t denotes the signal level at t time. Obviously, X (t+1) means the signal at t+1 time. In the process of reasoning, if the signal is "1", it equals to *True* (T); if it is "0", it equals to *False* (F). This feature can be modeled in abstract as the following properties.

Property 1:
If *reset* is T at time t, then the value out at *dataout* and *strobe* at time t are both F, i.e.
$\forall t.\ reset\ t \Rightarrow (dataout\ t=F) \wedge (strobe\ t=F)$ (1)
where $\forall t$ *means for all t;* $\Rightarrow$ *means implication.*

As described in SpaceWire standard , the data values are transmitted directly, the relation between data and strobe signal presents the fact that the output of the data signal should be 3 clock cycles later than the input. Therefore, this can be expressed in property 2:

Property 2:
If *reset* is T at time t+1 or t+2 or t+3, then the value output on *dataout* at time t+3 is F,

otherwise it is equal to the value input at time t on *datain*, i.e.

$\forall t.\ dataout\ (t+3) = if\ reset\ (t+1) \vee reset\ (t+2) \vee reset\ (t+3)\ then\ F\ else\ datain\ t$ (2)

Property 3:

If reset is T at time t+1, then the value output at *strobe* at time t+1 is F, otherwise, if *dataout* is equal at time t and time t+1, the value output at *strobe* at time t+1 is equal to the negation of the value at time t on itself otherwise the value at time t on itself, i.e.

$\forall t.\ strobe\ (t+1) =$

$if\ reset\ (t+1)\ then\ F\ else$

$(if\ dataout\ t = dataout\ (t+1)\ then\ \neg(strobe\ t)\ else\ strobe\ t)$ (3)

where ¬ means negation.

To state the specification property 3 in an intuitive logic expression, Xor operation is defined.

Definition 1: $\forall a\ b.\ Xor\ a\ b = \neg a \wedge b \vee a \wedge \neg b$

Therefore, (3) can be expressed using Xor as follows:

$\forall t.\ strobe\ (t+1) =$

$if\ reset\ (t+1)\ then\ F\ else$

$Xor\ (strobe\ t)\ \neg Xor\ (dataout\ (t+1))\ (dataout\ t))$ (4)

2.2 DS Encoding Implementation

The SpaceWire Encoding Circuit is designed with VHDL code. Only the process related to encoding is considered. For example, instead of *TX_ShiftReg*, *Datain* signal indicates the data transmitted. The simplified code is given in Figure 2:

```
IF (Clock'EVENT AND Clock = '1') THEN
    IF (Reset = '1') THEN
        D1          <= '0';
        D2          <= '0';
        DataOut     <= '0';
        St          <= '1';
        Strobe      <= '0';
    ELSE
        D1          <= Datain;
        D2          <= D1;
        DataOut     <= D2;
        St          <= St XOR NOT (D1 XOR D2);
        Strobe      <= St;
      END IF;
END IF;
```

Figure 2 Simplified code based on VHDL code

The approach to translating VHDL code into logic expression is not so straight. In addition, the proofs depend on the proper modeling and analysis of complex timing behavior. The tactics for modeling VHDL code in abstract have never been found in any referable paper or book. The method to model the VHDL code in behavior level is based on functions. Each function or any operation is modeled with predicates. All predicates are joined together with AND operator as shown in Figure 3. In such a way, the entire circuit can be modeled.

The predicate ONE is that for all times t the value of *out* is T.[2]
Definition 2: $\forall out.\ ONE\ out = \forall t.\ out\ t = T$

As known, has constructs to handle the parallel behavior designs, the value of the signal variable in the right-hand-side of the assignment statement is the one at the preceding time. For the purpose of expressing this behavior, REG is defined. The value of the output at time t+1 is that of the input at the previous time t, except at time 0. Since Time 0 refers to the exact time point that the device is power on, REG outputs F. [2]
Definition 3: $\forall inp\ out.\ REG\ (inp,out) = \forall t.\ out\ t = if\ t = 0\ then\ F\ else\ inp\ (t - 1)$

Similarly, *`IF…THEN…ELSE…`* can be modeled with predicate MUX. The input *sw* selects which of the other two inputs are to be connected to the output *out*.
Definition 4: $\forall sw\ in1\ in2\ out.\ MUX\ (sw,in1,in2,out) = \forall t.\ out\ t = if\ sw\ t\ then\ in1\ t\ else\ in2\ t$

The NOT gate is used to denote the situation that the value of *out* is always the negation of the value of *inp*.
Definition 5: $\forall inp\ out.\ NOT\ (inp,out) = \forall t.\ out\ t = \neg inp\ t$

Xor operation is modeled with a predicate XORING. This predicate states the case that the value of *out* is always *in1* XOR *in2* having no delay.
Definition 6: $\forall in1\ in2\ out.\ XORING\ (in1,in2,out) = \forall t.\ out\ t = Xor\ (in1\ t)\ (in2\ t)$

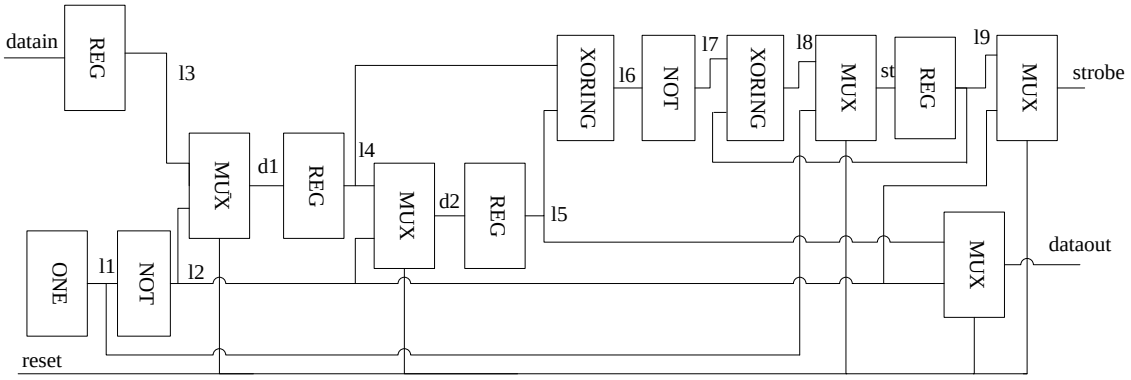


Figure 3 the connected predicates

Then *reset, datain, dataout, strobe, d1, d2* and *st* represent *Reset, Datain, Dataout, Strobe, D1* and *D2* in VHDL code, respectively. For the convenience of expression, *l1,*

l2, l3, l4, l5, l6, l7, l8 and *l9* are assumed as the temporary signal variables. The connected predicates are illustrated in Figure 3.

3 RESULT

Although the DS encoding circuit is small, it is worth doing verification on it because of its widely applications (The DS encoding scheme is also used in the IEEE Standard 1355--1995[4] and IEEE Standard 1394--1995 (Firewire) [5]). By applying the corresponding tactics and tacticals in HOL tool, the result showed the goal is proved, which suggested that the implementation of this circuit in FPGA hardware can imply the specification of this encoding scheme. So the design of the circuit can implement the behavior specified in the standard.

On the other hand, the efforts of the interactive verification work in a system might be huge. Our work compromises the merits of formal verification, simulation and testing. It suggests that a feasible scheme for verifying a safety system is applying theorem proving method on the critical and model checking method on the other parts.

4 CONCLUSION AND FUTURE WORK

By means of theorem proving with the aid of HOL tool, SpaceWire encoding circuit was verified and the result shows that the implementation is equivalent to the specification. Describing the behavior of the circuit with a general abstract model, it overcomes the limitation of simulation and testing. Furthermore, a new approach is proposed and proved to be usable in our work for modeling the VHDL code in abstract. More important, formal verification is expected be introduced into the process of design, since it is more helpful to ensure the design at the early stage of the product.

The successful verification on this small circuit is the first step of our investigation on applying the formal methods on the safety system. In the future, to verify some more circuits designed based on SpaceWire standard is our main aim so as to providing strong support to the correctness of our design in SST project.

Another plan for our future research is that more efforts will be done on investigating a compromise scheme for verifying the whole design applying formal methods in order to ensure the design in a critical security completed system.

5 REFERENCES

1. SpaceWire - Links, Nodes, Routers and Networks, ECSS Standard ECSS-E-ST-50-12A

2. The HOL System TUTORIAL (For HOL Kananaskis-4) , January 2, 2007

3. Michael C. McFarland, "Formal Verification of Sequential Hardware: A Tutorial", IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems, VOL 12, NO. 5, May 1993

4. IEEE Std 1394-1995, IEEE Standard for a High Performance Serial Bus, 1995, ISBN 1-5593-7583-3

5. IEEE Std 1355-1995, Standard for Heterogeneous InterConnect (HIC), 1995, ISBN 1-55937-595-7

IMPLEMENTATION OF A SPACEWIRE INTERFACE AS A COMPONENT WITHIN THE SPACE TELECOMMUNICATIONS RADIO SYSTEM (STRS) ARCHITECTURE

Session: Onboard Equipment and Software (Poster)

Long Paper

James Lux, Ryan Stern, Minh Lang, Gregory Taylor
Jet Propulsion Laboratory, California Institute of Technology
Pasadena California USA

E-mail: james.p.lux@jpl.nasa.gov

ABSTRACT

The SpaceWire IP core developed by NASA/Goddard Space Flight Center has been incorporated into the JPL-Software Defined Radio (SDR) being prepared for flight on the International Space Station as part of the CoNNeCT project. The JPL-SDR provides reprogrammable capability for the SPARC CPU and for the Xilinx FPGAs to implement present and future communications waveforms and networking functions. The implementation follows the new NASA Space Telecommunications Radio System Architecture Standard, which provides for abstracted interfaces among the various software and hardware components within the radio. The STRS architecture defines an Operating Environment (STRS OE), which provides basic platform capabilities and abstracted views of the functional infrastructure of the hardware/software platform. The JPL STRS OE provides SpaceWire services to user applications (i.e. SDR waveforms) implemented within the FPGA, reducing the need for a waveform developer to be aware of the details of SpaceWire implementation: it is just another data source or sink. The JPL implementation provides an RTEMS device driver to manage the SpaceWire cores and also exposes an interface to software running in the SPARC as well as to the more traditional modulator/demodulator functions implemented in the FPGA.

Time Distribution using SpaceWire in the SCaN Testbed on ISS

Session: SpaceWire Missions and Applications

Short Paper

James Lux

Jet Propulsion Laboratory, California Institute of Technology

Pasadena, California, USA

Dale Mortensen, Eric Anderson

NASA Glenn Research Center

Cleveland Ohio, USA

E-mail: james.p.lux@jpl.nasa.gov

ABSTRACT

SpaceWire is being used for time distribution and synchronization among the 3 different software defined radios (SDRs) and the avionics payload on the SCaN Testbed which is being produced by the CoNNeCT project for installation on the International Space Station in 2011. The SCaN Testbed will provide a platform for experiments on communication, networking and navigation using SDRs in space. SpaceWire timecodes will provide the means to transfer synchronization signals from waveforms (applications) implemented in one radio to another radio or to the controlling avionics system. Other messages needed for synchronization (e.g. "at the tone, the time is") will be carried either by the SpaceWire links or by MIL-STD-1553B command, control and telemetry interfaces. One of the radios, the JPL-SDR, provides abstracted synchronization interfaces to waveform components instantiated in its FPGAs, such as the code epoch or 1pps messages from a GPS, or other synchronization derived from S-band communications signals. Ground test interfaces provide sources and sinks for messages and time codes to allow calibration and testing.

Elastic Flow Control and Parallel Switch Design for SpaceWire Router

Session: SpaceWire Networks and Protocols (Poster)

Long Paper

Chunjing MAO, Yong GUAN

College of Information Engineering, Capital Normal University, Beijing, 100048, China/ Department of Computing, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong, China

Zili SHAO

Department of Computing, The Hong Kong Polytechnic University,

Hung Hom, Kowloon, Hong Kong, China

Jie Zhang

College of Information Science & Technology, Beijing University of Chemical Technology, Beijing, 100029, China

E-mail: mcjing@163.net, gxy169@sina.com, cszlshao@comp.polyu.edu.hk, jzhang@mail.buct.edu.cn

ABSTRACT

In a SpaceWire network, to connect equipments together, a SpaceWire router uses the wormhole routing to deliver packets. However, in the wormhole routing, there inherently exists the braking-problem that increases the average non-blocking latency. In this paper, we propose an elastic flow control mechanism to solve this problem. In addition, we propose a novel parallel switch architecture with pipelining to improve the transmission speed and switching efficiency, and optimize its FPGAs implementation. We implement our technique and test it in Xilinx FPGAs. The results show that on average the non-blocking latency can be reduced to 245ns at 200MHz. By using our elastic flow control, a long physical connection between two nodes can be established without reducing the bandwidth utilization. With our pipelined parallel switch architecture, the transmission speed can be enhanced to over 300Mbps and the circuit scale can be reduced 50% compared with the original design in FPGAs implementation.

1 WORMHOLE ROUTING

Wormhole routing is an effective solution for packet routing [1][2]. Each packet contains a header which holds the destination node address. As soon as the header for a packet is received, the router determines the output port to route the packet to by checking the destination address. If the requested output port is free, the packet is routed immediately to that output port. That output port is now marked as busy until the last character of the packet has passed through the router – indicated by the end of packet marker being detected by the router. Wormhole routing only cuts down on the amount of buffering used within each router and the delay for packets deliver.

Compared to a store and forward technique, where an entire packet is first received and stored before it is sent out of the router, NoC (Network on Chip) [32] can be realized. And delay of a single packet T_{lower} can be described as:

$$T_{lower} = (L_f / B_{ch}) \cdot D_{sd} + L_p / B_{ch} \qquad (1)$$

L_f is the length of each flit, D_{sd} is the distance between source node and destination node, B_{ch} is the bandwidth of the channel, and L_p is the length of packet. If $L_f << L_p$, the influence from D_{sd} to T_{lower} can be ignored.

Wormhole routing [1] is illustrated in Figure 1 which shows a packet being sent from one node to another through a routing switch (router). The header of the packet is marked as black, while the rest of the packet is marked as grey. As soon as the router receives the header, it checks the requested output port. If the output port is free, then the router makes a connection between the input port and the output port. The packet then flows through the router. When the end of packet (EOP or EEP) marker is received by the switch, the router terminates the connection and frees the output port for the next packet, which can come from any input ports.

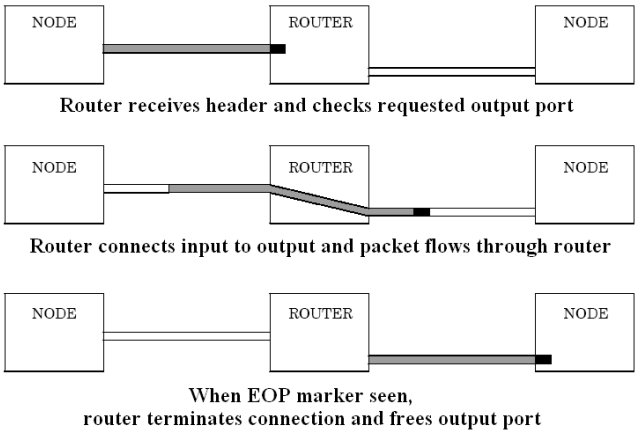


Figure 1. Wormhole Routing

Using of blocking flow control mechanism in wormhole routing, there is an inherent braking-problem [4][5]. When a header flip reaches the router, the flip has to wait until next channel is free. Transmission time of packet from header flip to end flip is T_b, and it can be expressed as:

$$T_b = T_{AD} + T_{ARB} + M \times T_s \qquad (2)$$

T_{AD} is coding time of address coder; T_{ARB} is arbiter time, and T_S is blocking signal transmission time in router. Because all the data flips are transmitted with pipelining mode, clocking time T_c need to be greater than T_b (see Equation 3). Otherwise, packet-loss would happen.

$$T_c > T_{AD} + T_{ARB} + M \times T_s \qquad (3)$$

Latency built up in router is with millisecond. A signal transmission can be finished with nanosecond, and clocking time is in millisecond. From Equation 3 we can see

that, working-frequency is limited by the braking-problem, and working-frequency is one of the key factors to improve the transmission rate.

2 ELASTIC FLOW CONTROL

Based on analysis above, an elastic buffer is applied to solve the brake problem in wormhole routing. Elastic flow control uses flip as the basic unit. In this paper, we propose an elastic flow control mechanism to illustrate the idea of design elastic buffer.

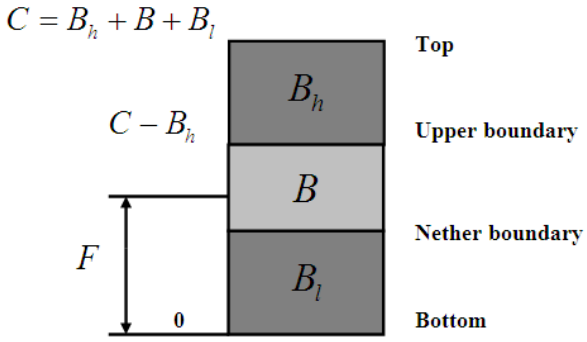


Figure 2. Structure of Elastic Buffer

As is shown in Figure 2, the capacity of buffer is C, the upper boundary is B_h, and the lower boundary is B_l. F is the number of bytes stored in a buffer, and $F = 0$ initially. While outgoing transmission rate is equal to incoming transmission rate, F maintains a fixed value. While outgoing transmission rate is lower than incoming transmission rate, the value of F will gradually increase. When it reaches the upper boundary, a flow control signal is built to notify the incoming node to stop transmission, and here remain B_h unused bytes in buffer for storing the incoming data in braking time. The braking distances are determined by B_h. By increasing B_h, braking distances can be prolonged.

Accordingly, the setting of B_l is to prevent flow breaking while read-out from buffer. While the value of F reaches nether boundary, flow control signal can be revoked to notify incoming node continuing transmission. It needs time for low control signal to be revoked from incoming node, and with B_l, it can prevent flow breaking when low control signal be revoked. Thus transmission would be more effective.

By using elastic flow control, address coder time T_{AD} and arbiter time T_{ARB} do not need to be calculated, and delay can be reduced by paralleling data transmission and channel switch. The flow control signal is built and revoked at the same time, which is equal to blocking time T_B. So T_c should be:

$$T_c > T_B + T_s \qquad (4)$$

It is easy to build a flow control signal, which can be finished within several nanoseconds. Compare with Equation 3 and Equation 4, braking-problem can be solved with elastic flow control. Also, there can be a long physical connection

between two nodes without reducing the bandwidth utilization by B_h and B_l in synchronous mode.

3 PARALLEL SWITCH ARCHITECTURE

Normally, there are three steps for a packet passing from input port to output port in SpaceWire router:

A. Reading header for the packet and sending the destination node address to routing table

B. Finding destination node address in routing table and to decide the output port of the packet

C. Sending the packet to the decided output port

There is a read/write competition when more than two input ports sending packet to the same output port. This would trigger the arbitral mechanism in router. After arbitration, the input port with higher priority can send its packet. That is, only one input port can send packet at a certain time; others would sending packet sequentially by arbitral result. Transmission efficiency of router would be confined by this serial arbitration mode.

We design a pipeline based SpaceWire router to improve the transmission efficiency. The objective is to design a pipeline based non-blocking parallel switch, as is shown in Figure 3.

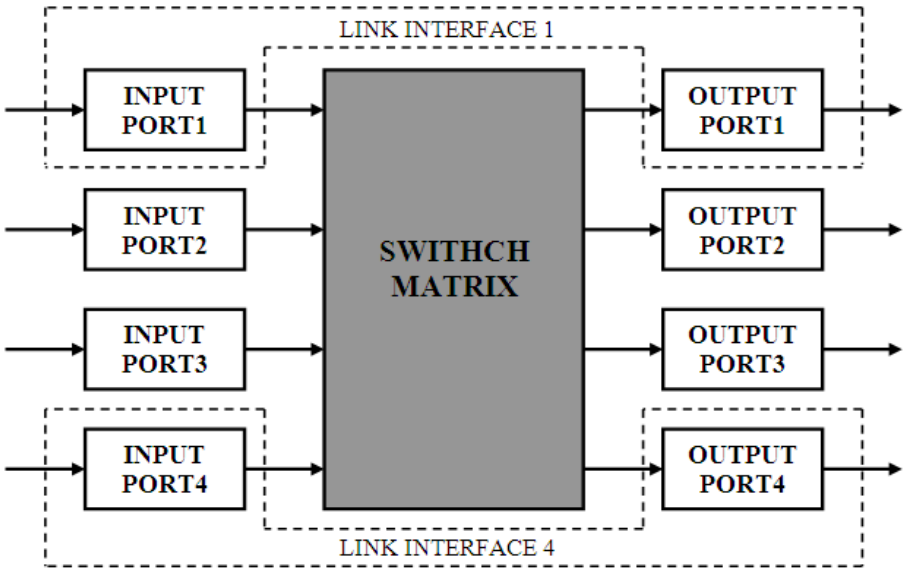


Figure 3. Illustration of Switch Matrix

Circuit of packet header detection and packet reorganizing can be very complex with a large circuit scale in original SpaceWire router design. We propose a transaction processing pipeline for packet header detection and packet reorganizing, which can reduce the complexity of circuit design and power consumption, and improve the reliability of SpaceWire router. The architecture of parallel switch pipelining is illustrated in Figure 4. We realize this architecture with FPGAs, and experiment

results show that, by using transaction processing pipeline, the circuit scale can be reduced about 50% compare to original design.

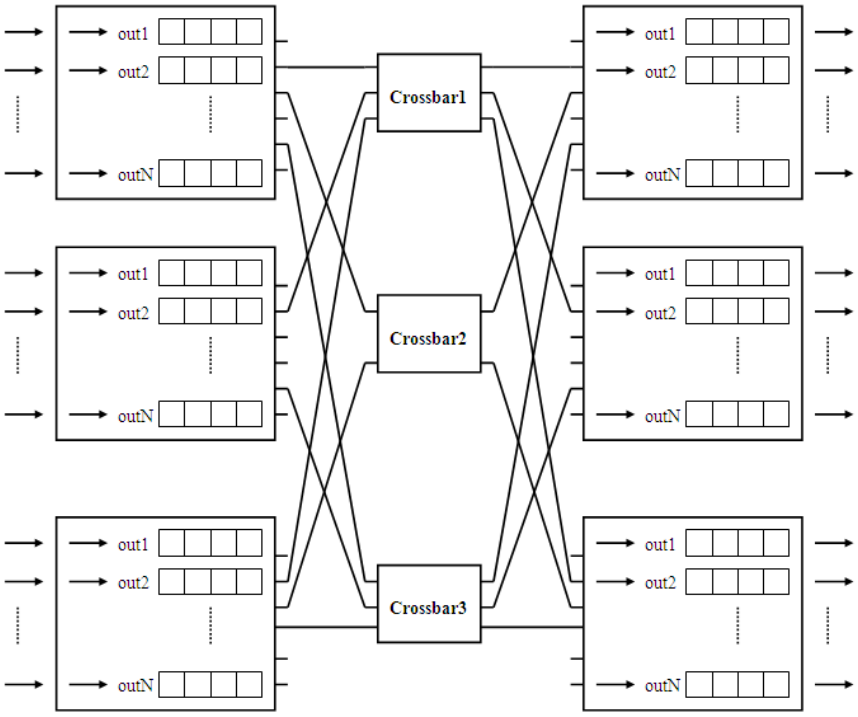


Figure 4. Architecture of Parallel Switch Pipelining

As illustrated in Figure 4, this architecture composes of K no-buffer crossbar switches and M input sharing storage modules. There are V external interfaces for each input/output sharing storage module, which connects to all crossbar switches.

There are $M \times V$ VOQs (Virtual Output Queue) in each input sharing storage module, which stores packets to different destination. V output queues in each output sharing storage module store packets wait be sent. In each arbitration cycle, input port picks up M queues from $M \times V$ VOQs randomly, and submits scheduling requests to switch. Switch services each queue by polling and feedbacks authority information to input port by scheduling result. The first packet of VOQs that appointed by authority information will pass switch to output queues, and it will be sent to external links after message reorganization. Push-reverse mechanism can be adopted to avoid overflow in output.

In case of multi-input sending packets to one output, we adopt pipeline technology for multi-transmission by time-sharing operation. Also, routing table can copy to each input port, and there is no need for arbitral mechanism. While a packet arrives in input port, it decides the output port by finding in routing table by header of packet, and it will be sent the packet. While multi-input transmission, it would assign a time-token for each input port. Packet can be transmitted when time-token of this input port is enabled, and transmission is stopped once time-token is disabled. All incoming packets can follow this way. Assigning time-token can be controlled by routing algorithm. The basic idea is to assign time-token to each input port alternately by using pipeline, and routing algorithm can be optimized in some specific application. It

would improve transmission performance in SpaceWire network, and accessing for each input/output port can be achieved simultaneously.

4 EXPERIMENTS

We conduct experiments by building a platform with Xilinx FPGAs, which is shown in Figure 5. The platform consists of two routers, and each router is connected with several nodes. Each node includes the data source of video, image, audio, and instructions. Each router connects with a PCI node, which is used to connect the host PC. The host PC can observe and configure the router via PCI node.

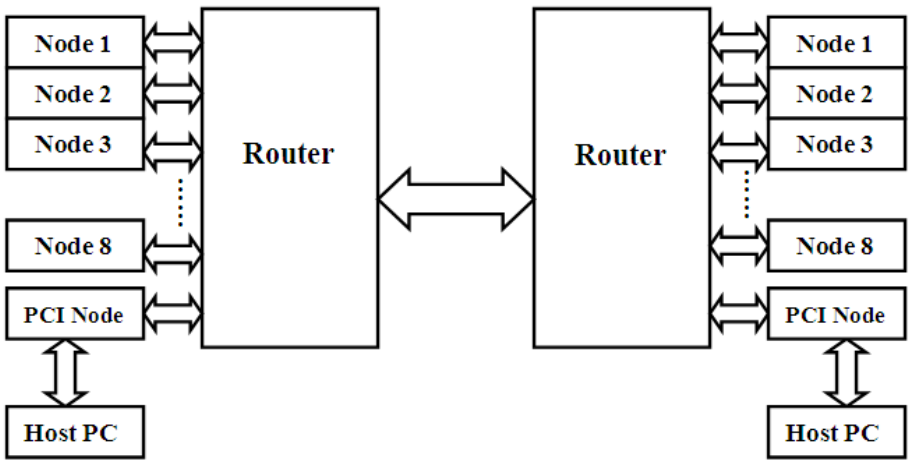


Figure 5. Illustration of Experiment Platform

The experimental results show that on average the non-blocking latency can be reduced to 245ns at 200MHz. By using our elastic flow control, a long physical connection between two nodes can be established without reducing the bandwidth utilization. With our pipelined parallel switch architecture, the transmission speed can be enhanced to over 300Mbps and the circuit scale can be reduced 50% compared with the original design in FPGAs implementation.

5 CONCLUSION

In this paper, we solve the braking-problem by using elastic flow control mechanism. We proposed novel parallel switch architecture with pipelining to improve the transmission speed and switching efficiency, and we optimize its FPGAs implementation. The experimental results show that on average the non-blocking latency can be reduced to 245ns at 200MHz, and transmission speed can be enhanced to over 300Mbps. The circuit scale can be reduced 50% compared with the original design in FPGAs implementation.

REFERENCES

1. “SpaceWire protocols”, ECSS-E-ST-50-11C, July 2008

2. “SpaceWire standard - Links, nodes, routers and networks”, ECSS-E-ST-50-12C, 31 July 2008

3. Ai Zhen. "Design of Wormhole Routing Based on Black-bus," Master Degree Thesis of University of Electronic Science and Technology of China, 2007(5).

4. An Xuejun, Zhu Faming, Gao Wenxue, Wu Dongdong. "The Design and Implementation of the Elastic Buffer in Wormhole Routing Chips," Computer Engineering and Applications, 2002, 7.

5. Xiao Xiaoqiang, Jiang Yuqin, Jin Shiyao, He Hongjun. "BWR—Buffered Wormhole Routing Switching," Chinese Journal of Computer, 2001, 24(1).

VCOM: CONTROLLING A REMOTE RS232 INTERFACE OVER SPACEWIRE

Session: SpaceWire Test and Verification (Poster)

Short Paper

Alex Mason

STAR-Dundee Ltd,

c/o School of Computing, University of Dundee, Dundee, Scotland, UK

Steve Parkes

University of Dundee, Dundee, Scotland, UK

E-mail: alex@star-dundee.com, sparkes@computing.dundee.ac.uk

ABSTRACT

The STAR-Dundee Virtual COM Port (VCOM) provides a serial interface from software on a Windows-based PC, via the standard COM port interface, to a remote UART, tunnelled over SpaceWire. It provides the ability to communicate with and test remote serial devices and applications in a lab environment. In this paper the features provided by the VCOM software, including remote UART configuration, and a protocol allowing serial errors and status to be propagated across SpaceWire are described. Applications of this technology to the configuration of a prototype wideband spectrometer, and potential extensions to the VCOM abstraction are also discussed.

1 OVERVIEW

The STAR-Dundee Virtual COM Port exposes a remote UART (universal asynchronous receiver/transmitter) on a SpaceWire Network [1] to a PC as a standard Windows COM port. A specially-designed Windows driver provides this driver/API interface. The virtual COM port driver may be used with any STAR-Dundee USB-based device.

The virtual COM port appears to Windows and may be interacted with in exactly the same way as a physical COM port using the Windows File and Communication functions. The system operates by placing serial traffic received from a UART into SpaceWire packets and transmitting them across the network to the PC. Similarly, SpaceWire packets of the correct form received at the UART node are transmitted in serial form (Figure 1).

When data is written to the virtual COM port, it has a SpaceWire logical address and protocol byte prepended, and is sent across the SpaceWire network. The VCOM driver continuously reads incoming SpaceWire data and buffers it, ready to be consumed by the user using the Windows File and Communication functions.

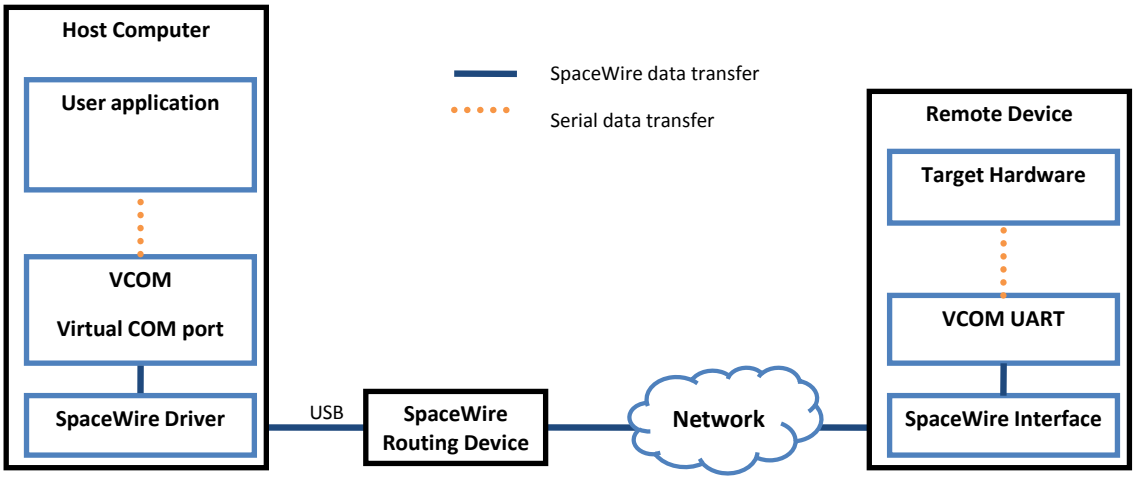


Figure 1: Overview of VCOM Communication

2 VCOM DRIVER

In order to appear to Windows as a COM port, the VCOM software must be implemented as a device driver.

2.1 UMDF

The VCOM driver makes use of the Windows User Mode Driver Framework (UMDF), which allows it to use the existing STAR-Dundee user mode API. A major benefit of using UMDF is that the VCOM driver runs in a different thread from the requesting process and has access only to the address space of the process in which it is running.

2.2 REMOTE UART CONFIGURATION

All Windows Serial IOCTLs (I/O controls) are handled by the VCOM driver, and the file type of the file handle returned when opening a VCOM port is a character file, enabling any legacy application that requires a COM port interface to be used with VCOM. RMAP (Remote Memory Access Protocol) [2] is used to perform all configuration operations on the UART over the SpaceWire link.

The SpaceWire Virtual COM port configuration API provides functions which can be called to configure both the virtual COM port device on the host PC and the remote UART attached via the SpaceWire link. A graphical configuration tool is provided which makes use of this API.

2.3 PACKET FORMAT

VCOM uses a simple packet format **(**Figure 2**)** containing the Protocol ID [3] and a status byte in addition to the data bytes. The protocol ID is the first byte after the logical address. The use of logical addressing allows multiple VCOM drivers on a single machine, each communicating with a different remote UART to share the same SpaceWire interface. The Protocol ID is used by the driver to ensure a valid packet has been received. ID 0xF0 has been chosen as it has not been assigned by the SpaceWire working group. The next byte in the packet is the status byte. This bitmask

provides information to the VCOM driver of any errors that have occurred at the UART, for example framing or parity errors, or whether a break state is in progress.

First Byte Transmitted				
Logical Address	**Protocol Id**	**Status Byte**	**Data**	**EOP**
0xAB	0xF0	0x00	...	...

Figure 2: VCOM packet format

2.4 FLOW CONTROL

Flow control between the host computer and the remote UART is performed using SpaceWire flow control. As the serial communication between the remote UART and the serial hardware is a 3-pin implementation (receive, transmit and ground) there is no flow control. Instead, a large buffer is present at the VCOM SpaceWire interface, and is used when receiving large numbers of packets in quick succession. When a 5-pin UART implementation is being used with the CTS and RTS signals available, flow control can be provided to the UART.

3 CURRENT APPLICATION OF VCOM

STAR-Dundee Ltd is currently developing a prototype wideband spectrometer instrument with the University of Dundee, EADS Astrium Ltd and RAL. This device samples an analogue signal at around 3 Gsamples/s and performs spectral analysis to extract a signal buried deeply in noise. The prototype instrument is targeted at atmospheric chemistry missions. A Xilinx FPGA is used to implement the prototype Fast Fourier Transform (FFT) with the FFT code itself being developed by Astrium.

The prototype wideband spectrometer is shown in Figure 3.

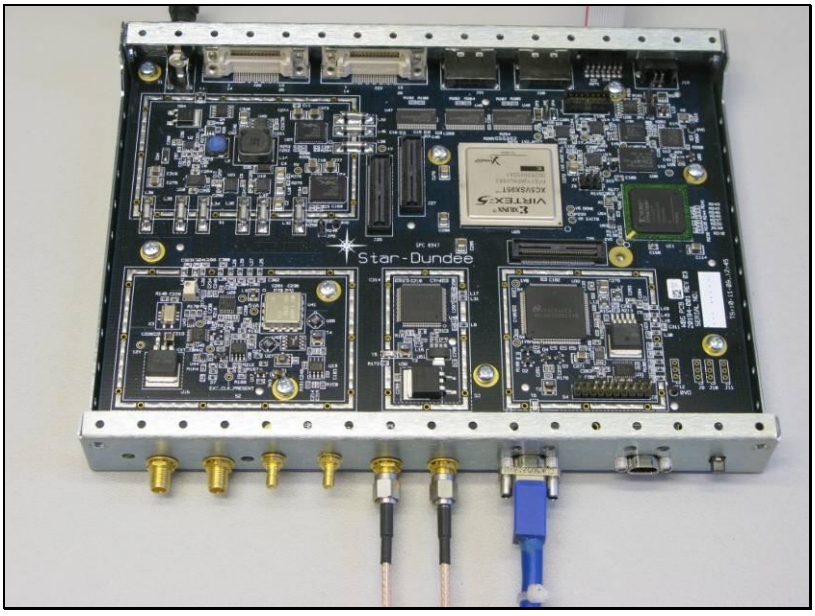


Figure 3: Prototype wideband spectrometer

To simplify integration in the development environment an RS232 interface was required for controlling the FPGA along with a JTAG port for programming it. However, using a single SpaceWire interface to configure and access the device is preferable. VCOM was developed to solve this problem.

4 Potential Applications of VCOM

The VCOM abstraction is not limited to controlling a remote UART. Keeping the same software and SpaceWire hardware, a VCOM core could be created enabling the remote UART to be replaced with *any* remote streaming interface. Instead of a remote UART, one could keep the same abstraction of a COM port, but control, for example, a generic FIFO, I^2C, Serial Peripheral Interface Bus (SPI), a Xilinx LocalLink bus [4], Altera Avalon Streaming Interface bus [5] or a Xilinx Fast Simplex Link bus [6] (Figure 4).

The Windows serial functions would still be used to control the data streamed over SpaceWire (i.e. start bits, stop bits, data length parity, timeouts), with additional IOCTLS used to allow configuration of specific streaming interfaces where necessary.

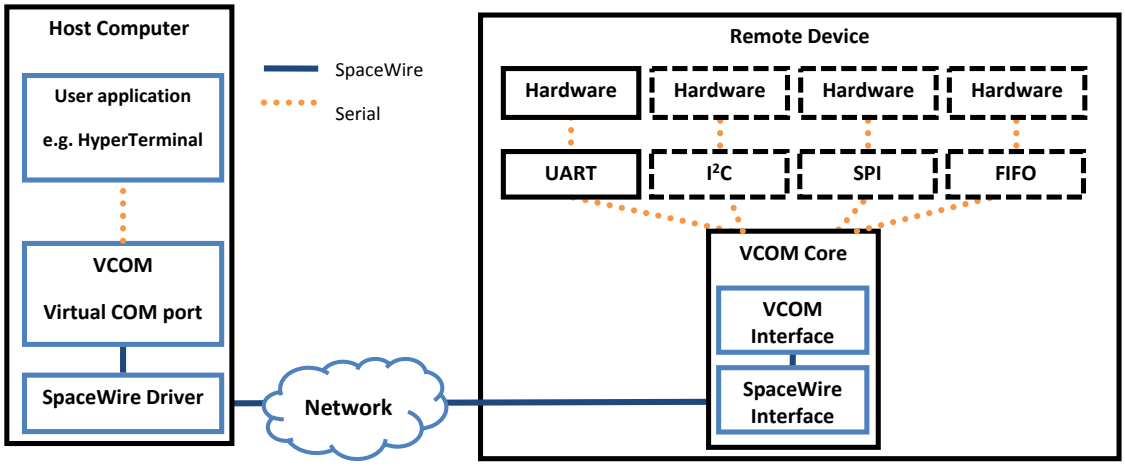


Figure 4: Extending the COM port abstraction for controlling any remote streaming interface

5 Summary

VCOM allows serial applications using the standard windows COM port to be used to access hardware over SpaceWire. This paper has shown an application of VCOM to allow communication with a prototype FPGA to be performed over SpaceWire. In addition this paper has highlighted further potential uses of the COM port abstraction for controlling remote hardware.

6 References

1. ECSS, “SpaceWire, Links Nodes, Routers and Networks,” European Cooperation for Space Standardization, ECSS-E-ST-50-12C, Issue 2, 2008.

2. ECSS, “SpaceWire – Remote memory access protocol,” European Cooperation for Space Standardization, ECSS-E-ST-50-52C, 2010.

3. ECSS, “SpaceWire protocol identification,” European Cooperation for Space Standardization, ECSS-E-ST-50-51C, 2010.

4. Xilinx, “LocalLink User Interface,” [Online]. Available: http://www.xilinx.com/products/ipcenter/LocalLink_UserInterface.htm. [Accessed: Apr. 05, 2010].

5. Altera, “Avalon Interface Specifications,” Apr, 2009. [Online]. Available: http://www.altera.com/literature/manual/mnl_avalon_spec.pdf [Accessed: Apr. 05, 2010].

6. Xilinx, “Fast Simplex Link (FSL) Bus (v2.11b) Data Sheet,” Jun, 26, 2009. [Online]. Available: http://www.xilinx.com/support/documentation/ip_documentation/fsl_v20.pdf. [Accessed: Apr. 05, 2010].

METHOD PROVIDING FAULT TOLERANCE OF SPACECRAFT ELECTRONICS BY N-MODULAR REDUNDANCY IN INFORMATION SPACE OF SPACEWIRE NETWORK

Session: SpaceWire Onboard Equipment and Software (Poster)

Short Paper

Alexander Popovich

Actel. ru, 212 Moskovsky pr., St. Petersburg, Russia,196006

E-mail: popovich@actel.ru

ABSTRACT

An approach to practical implementation of fault-tolerant redundant onboard computer system is suggested. Redundancy of electronics modules (microcomputers) is achieved via triple majorization of outgoing network packets in SpaceWire network. Group of modules is self-synchronised by SpaceWire network tools in compliance with patented procedure. Information is recovered and synchronised after single failures by protocol ECSS-E-50-11 SpaceWire/RMAP utilities.

Method is applicable for microcomputers with processors built both as IP cores in FPGA (for example, Intel 186 or Leon3) and single microcircuits (RAD750). To implement the method IP core of SpaceWire router for Actel FPGA with extended services of packets majorization, support of data recovery and synchronisation protocols is designed.

To test the method a demonstrator of space vehicle full-scale onboard network with two SpaceWire channels and redundant computer is developed and manufactured.

Satellite electronics reliability requirements are typically very high due to extreme conditions onboard: radiation, huge thermocline and mechanical shocks at the launch. The most common approach to provide high reliability is triple redundancy for almost all functional electronic modules. Synchronous operation of modules is essential to organise hardware voting schemes so it has to be provided by certain software and/or hardware solutions. Modern high speed networking technologies are capable to solve both synchronisation and N-modular redundancy voting tasks inside virtual information space. The approach for virtual N-modular redundancy described below is based on an approved patent application (RU 2008146151/09(060313)) regarding procedure and setup for self-synchronisation of modules by SpaceWire network tools. Full compliance of the method with existing international networking standards[1] opens the way for its practical application.

[1] Standards: ECSS-E-50-12C SpaceWire and ECSS-E-50-11 SpaceWire/RMAP

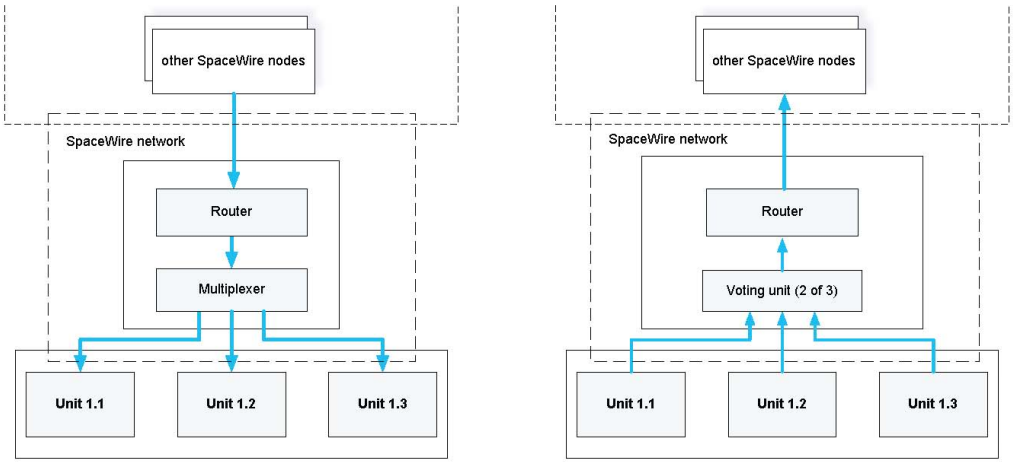


Figure 1: Solution outline

The most practically recognised N-modular redundancy system is for N=3. Triple redundancy solution for CPU-based SpaceWire node is outlined at Figure 1. The router operation is modified both for incoming and ongoing data transactions. Incoming data packets are forwarded to all three units in the node simultaneously by custom multiplexer. Provided all of three identical units of the node are operating synchronously their outgoing data packets may be majorized and "2 of 3" voting principle may be applied. If one of the three packets is corrupted or absent the rest of the network will be not affected. It's an important note that all of three units in the node may avoid any data exchange between each other in normal mode so any of these units "may presume" its uniqueness. This may simplify software design considerably.

For the system to operate data packet majorization should be performed within pre-programmed time frame Tmaj which is practically data propagation delay for the voting scheme as shown at Figure 2. Tmaj is also related to maximum asynchronism of unit operation within the redundancy scheme.

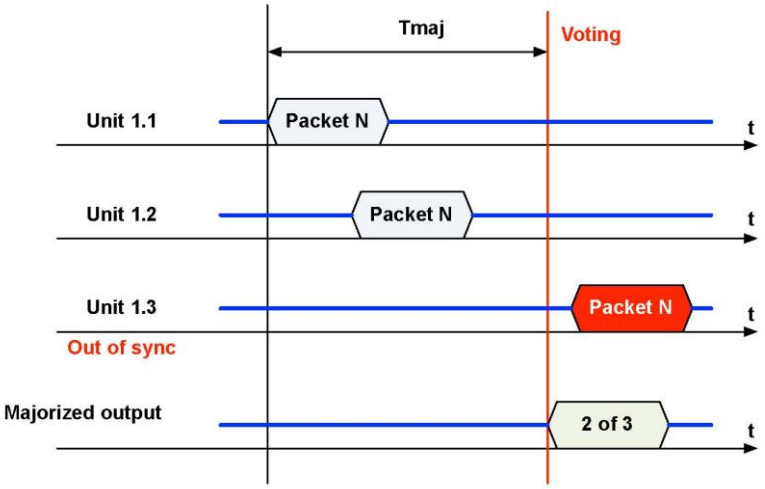


Figure 2 :Data packet majorization

Once any of units is late to provide correct packet within Tmaj or sends the data not similar to other's an error is detected but the node in general is still fault-less and the rest of SpaceWire nodes are not affected. As real-world clock generators provide variable frequencies the units will run same software/algorithms with slightly different speeds and special efforts must be taken to keep system synchronism.
A suggested approach to synchronise all units within a node is based on the local network broadcasts to allow unit processors operation for a pre-defined number of clock cycles. Once all processors of the node perform similar number of clock cycles within a certain time frame the synchronism precise enough for data majorization scheme to operate may be maintained.

To implement the principle each of units within a node must have a dedicated timer for synchronisation. On the count end an interrupt has to be generated to perform a sync packet broadcast by each of the units. Once the packet arrives (propagated to all units at the same time) all timers has to restart and late sync packets to be ignored as shown at Figure 3.

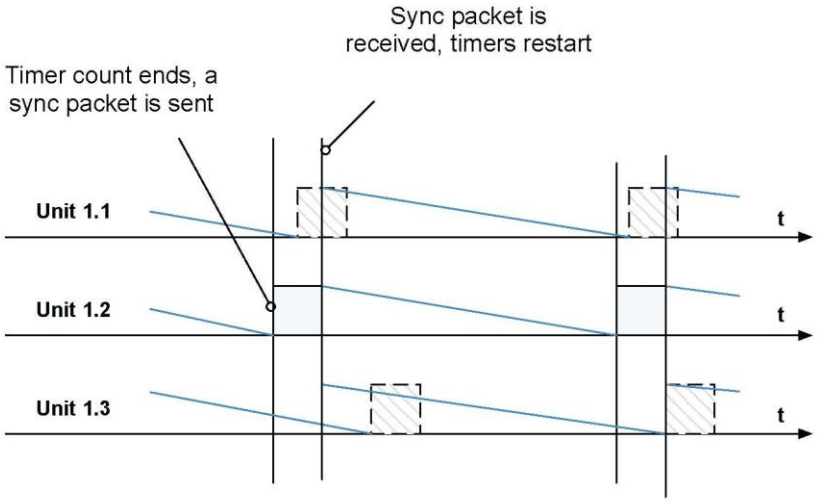


Figure 3 :Synchronisation principle

Its essential that all units perform same number of instructions (clock cycles) between the synchronisations. This may be achieved by halting the processors within every synchronisation periods as shown at Figure 4. Effective running time T2 will be less then sync time T1 but practically the performance loss will not exceed 1% as may be easily shown by calculations. For normal operation the standby time T1-T2 must be bigger than possible time difference between system's slowest and fastest clock generators to count the necessary number of cycles. With typical clock precision dispersion of 1 ppm the standby time T1-T2 will not affect system performance. The overall operation scheme is shown at Figure 5.

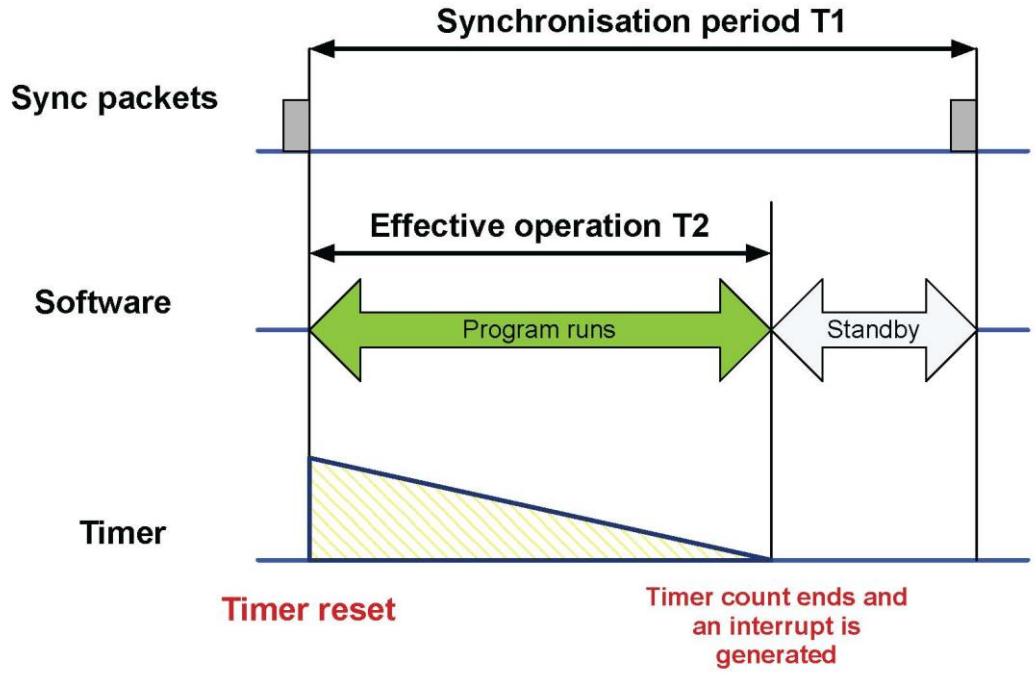


Figure 4: Operation of the synchronised unit

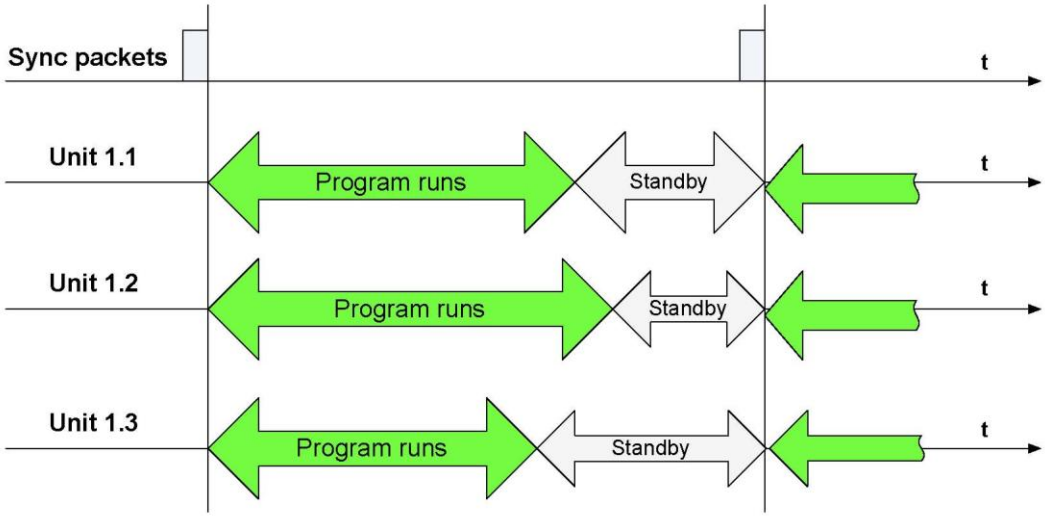


Figure 5: Operation of three units within a node

Simplified structure of FPGA implementation for the unit being discussed is presented at Figure 6. Two major functional blocks are: modified SpaceWire controller and processor unit. Processor unit may include almost any CPU IP ranging from 8051 up to Cortex or Leon3. Also external processor may be used once its interface supports standby mode predictable start/stop operation. RMAP protocol support may be implemented for system recovery after errors and for initial synchronisation as a software-free method to fill data memory of the unit with information.

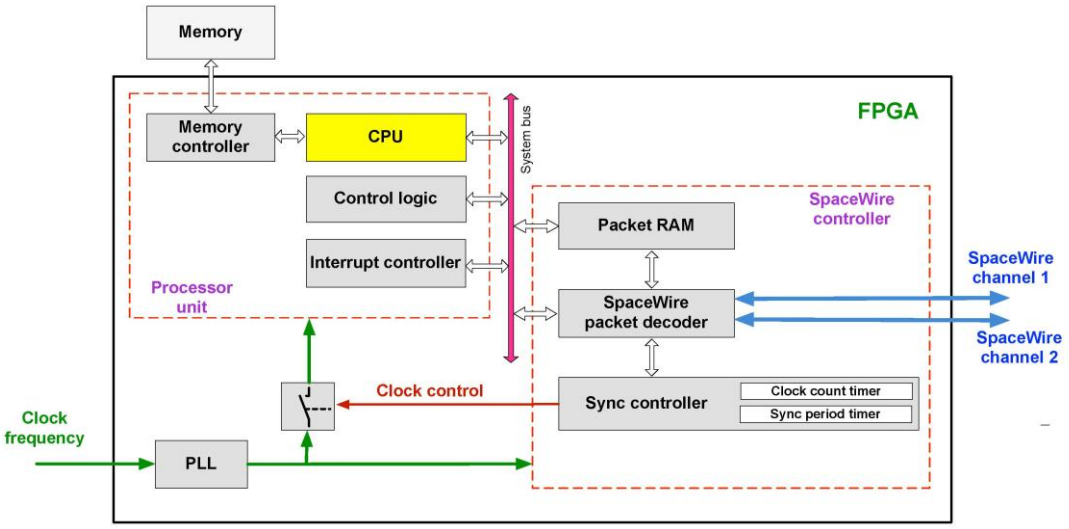


Figure 6: FPGA-based unit structure

As no ASICs are currently available on the market to implement the method described above, the only way possible is to use FPGA. The radiation-tolerant RTAX family Actel FPGAs with hardware TMR of each logic tile may be used both for unit design and for router design.

There is also a chance to use SpaceWire-based triple redundancy system for non-SpaceWire systems as shown at Figure 7. Three identical processing units with integrated CPU IP cores and SpaceWire controllers may be placed on the same PCB. Only two additional FPGA-based routers/bridges are necessary to organise local SpaceWire network and implement the method as discussed. The approach may be less power and space consuming compared to multi-layer TMR of the hardware signals within the same PCB.

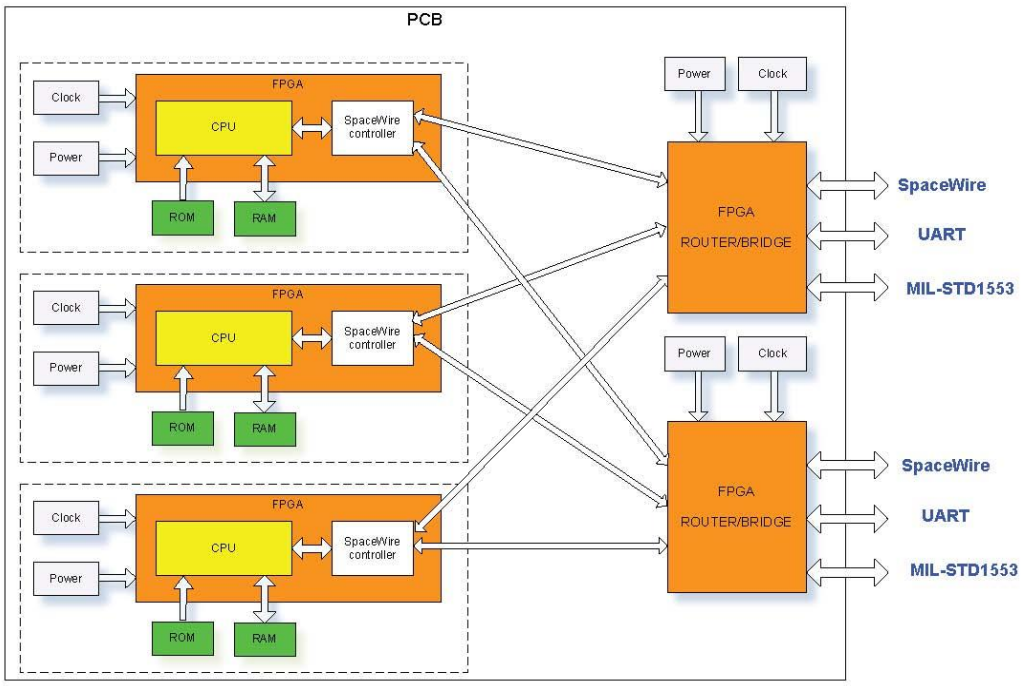


Figure 7: Single-board solution for SpaceWire-based TMR in non-SpaceWire environment

A full-function demonstrator based on Intel 186 compatible CPU IP cores was designed to test the method featuring the idea, the SpaceWire network and Actel FPGA.

RECONFIGURABLE IP-BLOCK OF TERMINAL NODE CONTROLLER

Session: SpaceWire Components (Poster)

Short Paper

F. V. Schutenko, E. A. Suvorova, A. Bayda

St. Petersburg State University of Aerospace Instrumentation
67, Bolshaya Morskaya st. 190 000, St. Petersburg RUSSIA

E-mail: suvorova@aanet.ru,

Vladimir Goussev, Petr Gussarov, Konstantin Bragin

SIC "ELVEES", Moscow

E-mail: vgoussev@elvees.com

ABSTRACT

In the paper we present reconfigurable IP-block terminal node controller (without processor core) that is based on the SpaceWire protocol. This IP-block includes a two-port SpaceWire routing switch and controllers for two SpaceWire transport protocols: RMAP controller unit and STP (Streaming Transport Protocol) controller unit.

RMAP controller in this IP-block is used for providing remote configuration of terminal node modes of operation. STP controller is used for data stream transfer to hosts.

The embedded in it small routing switch with two SpaceWire ports could be used for terminal node throughput increase, for building of daisy-chain interconnections of terminal nodes structures (if data flows from terminal nodes are not intensive), for fault-tolerant connection to SpaceWire networks.

Possibility of IP-block reconfiguration is considered. The RTL model of this IP-block could be configured as only RMAP or only STP controller. The SpaceWire routing switch external ports number could be vary from one to four.

In the paper we present main features of the designed protocol components, present some research results of RMAP and STP data transfer. We evaluate overhead and delays for same data flows translation with using RMAP and STP, compare power consumption by implementations of these protocols for same data flows transfers.

SPACEWIRE NETWORK TOPOLOGIES IN DISTRIBUTED DATA ACQUISITION AND CONTROL SYSTEMS

Session: SpaceWire Networks and Protocols (Poster)

Short Paper

E. A Suvorova

St. Petersburg State University of Aerospace Instrumentation
67, Bolshaya Morskaya st. 190 000, St. Petersburg RUSSIA

E-mail: suvorova@aanet.ru

ABSTRACT

In many applications SpaceWire networks are used for data transmission from sensors (sensor fields) to an onboard computer, for transmission from a computer to distributed actuators, data sinks. Multiplexing of data packet flows with small density to high density packet flow task and inverse task arise. In the article we consider problems of packet flow multiplexing with SpaceWire routing switches.

1 INTRODUCTION

The SpaceWire allows networks with arbitrary topology, ensures scalability to support various requirements in number of nodes, throughput and fault-tolerance. The article considers methodology for development of efficient SpaceWire interconnection topologies for distributed onboard data acquisition and control systems.

One of SpaceWire interconnections problems is efficient support of data flows from multiple low intensive sources. Data packet flows from primary sources may have low density. SpaceWire links utilization could be very low, cabling and numerous routing switches to support many terminal nodes in interconnection could be excessive in implementation cost. To deal with the cost-efficiency problem for low throughput segments of SpaceWire networks daisy-chain topologies could be efficient.

Multiport SpaceWire terminal nodes application for daisy-chains implementation is considered. Summary data packet flow from such a chain could provide reasonable load for a routing switch port. In the article we estimate hardware costs and data packet delivery latency for hierarchical network structures with and without daisy-chains. Research results are corroborated by SpaceWire networks simulation.

2 DATA PACKET TRANSMISSION TIME

Let’s consider the tack of some SpW data sources (data flow is 1Mbit|c) connection to one destination host; system includes 15 sources and one host.. Data flows could be multiplexed with using SpW routing switches (Figure 2, a) or daisy-chain (Figure 2, b). In switch based system 15 ports could be used for source nodes connection and

one port – for data transmission to host or next level routing switch (if system includes some levels of multiplexing). In such system link rate from sources could be small for power reduction or because of long cables. Link rate to the host typically is essentially higher. Therefore the full packet buffering in input ports of switch that are connected to sources should be used, as it is illustrated by Figure 1: it allows excluding Idle symbols transmission from switch to host when next data symbol from current packet is not yet received from source, because rate of its link is low.

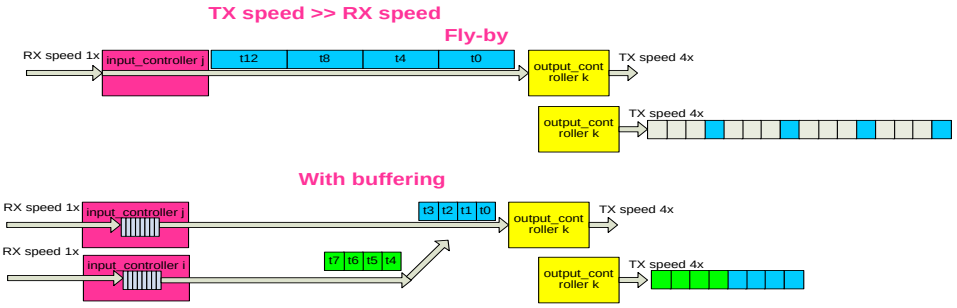


Figure 1

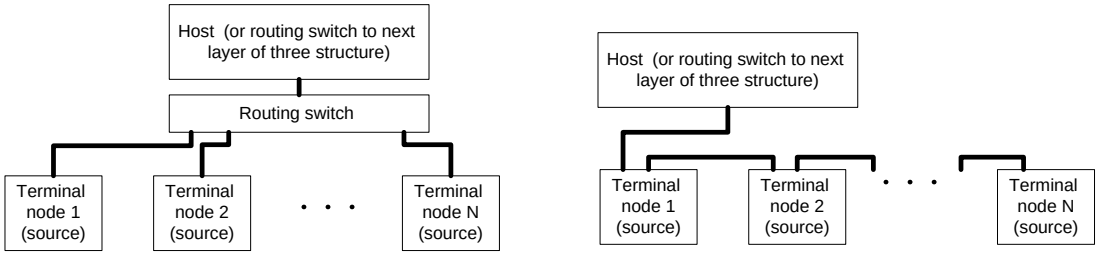


Figure 2

The maximal data packet transmission time from source node to host via switch (or next level switch) could be evaluated with formula (1) if data packet generation moments in different terminal nodes are independent random variables:

$$T_{\max sw} = T_{buf} + T_h + (N-1) * T_{trans} \qquad (1)$$

Here T_{buf} – time between start of packet transmission from source (from internal buffer of source if it exists) to the switch and receipt of last byte of packet into switch input port buffer. T_h – header processing time, depends on switch specific features. N – the number of nodes connected to this switch. T_{trans} –the transmission time from input port to output port of switch and then to host. It depends on data transmission rate via switch matrix, link to host rate and output port load. T_{buf} depends on link rate, source data flow intensity, packet length and terminal node and switch specific features. Two possible variants: 1) the packet is fully formed in node's buffer before sending out; T_{buf} depends only from the link rate; 2) the packet transmission from source to host begin when the source have prepared first packet symbol; next words are sent one by one when ready; T_{buf} depends on data generation rate in the source.

For daisy-chain (Figure 2, b) every source terminal node should include two ports routing switch SpW. The packet transmission time between terminal node and host (next level switch) could be evaluated by next formula:

$$T_{\max chain} = T_h + T_{trans} + (N-1)*(T_h + 2*T_{trans}) \qquad (2)$$

N is a serial node number in the chain; T_h – the header processing time in terminal node's switch; T_{trans} – the packet transmission time between output port of terminal node and input port of next terminal node (considered equal for all stages of chain).

Dependency between T_{max} and data packet length is presented on Figure 3. We consider first variant of packet forming. For all packets overhead (the length of SpW header and other data form transport and application level are equal 8 bytes in this example)

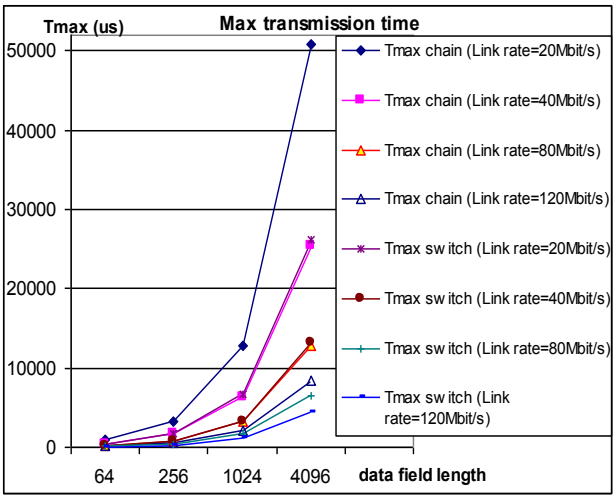


Figure 3

Diagrams show that maximal transmission time for a chain is about two times bigger than for a switch. For many applications these time is acceptable, but for some applications it could be too big.

For a switch based system transmission, when all sources generate data packets practically at the same moment, is the worst case – packet from one source should wait until packets from all other nodes would be sent o; all packets are received by switch in parallel but only one packet could be sent to output port in one time.

$$T_{stsw} = T_{\max sw} \qquad (3)$$

For daisy-chain system this case corresponds to the best variant. Every packet starts forwarding to neighbor node practically in one time. When packet arrives at next node it do not need to wait for transitions of any other packet; T_{stsw} is equal to $T_{stchain}$.

$$T_{stchain} = T_{\min chain} = T_{buffin} + T_h + T_{trans} + (N-1)*(T_h + T_{trans}) \qquad (4)$$

Dependency between average packet transmitting time and number of nodes in daisy chain when all packets are generated in one time are at Figure 4. (data field 64 bytes).

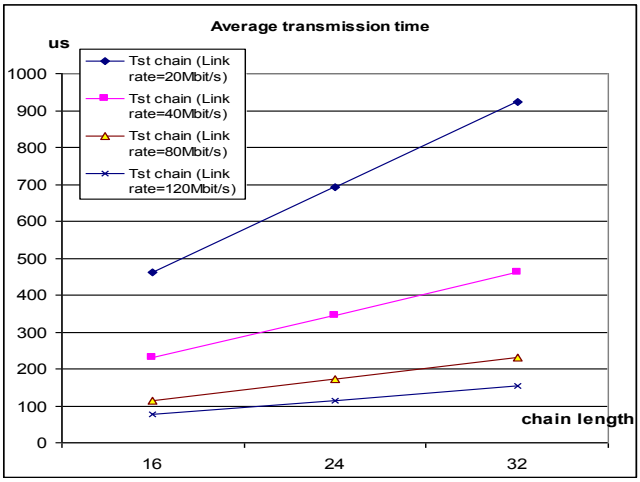


Figure 4

For a system, in which we need to collect information from number of nodes that is bigger than number of one switch ports, 3 basic variants of structure: fully switch based structure; structure in which at low level are daisy-chains and at next levels are switches; and structure fully based on daisy-chain. For example we consider system with 60 sources. This system could include 5 switches (4 switches multiplex information from sources - 15 to 1 each and one second level switch) (structure 1, **Figure 5**, a); 13 switches (12 switches multiplex information from sources - 5 to 1 each and one second level switch) (structure 2, **Figure 5**, a). System could include 4 daisy-chains with 15 nodes in each and one second level switch (structure 3, **Figure 5**, b); could include 12 daisy-chains with 5 nodes in each and one second level switch (structure 4, , **Figure 5**, b). System could be one daisy chain with 60 nodes (structure 5, , **Figure 5**, c).

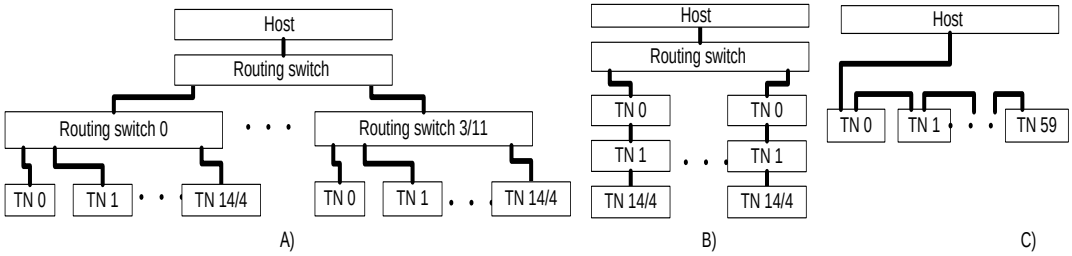


Figure 5

Dependency between average packet transmission time and link rates for these structures are presented on Figure 6 (data field size is 64). For structures 1, 3 and 2, 4 packet transmission times are practically equal. It shows that packet transmission time for all structures is practically equal when data transmission rate on low level is high. When data transmission rate on low level is low the best transmission time (in 2,5 times better than in other structures) are for structures 2 and 4 (in which on low level the switches with small number of ports and short daisy-chain are used).

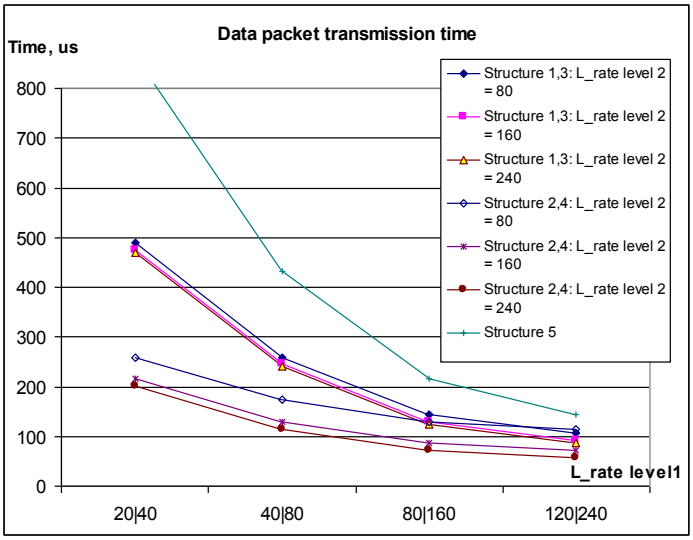


Figure 6

For long daisy-chain with 60 nodes average and the switch based system transmission time on high data transmission rate (240 Mbit/c) is practically equal.

3 HARDWARE COST

Typical structure of SpW routing switch includes SpW ports (codec SpW), SpW port's controllers (arbitration blocks, switch fabric channel controllers, data buffers), switch fabric, routing table and control|state registers. Hardware cost ratio of different component in switch essentially depends on RTL specific and technology libraries. But we select some basic ratios.

Let's compare hardware cost of 2-ports switch that need to be in terminal nodes for daisy-chain and 16-ports switch. The 16-ports switch includes 16 ports and controllers, one configuration port controller; 2-ports switch includes 2 ports and controllers, also hardware cost of these components of 2-ports switch is in 5,6 times less than for 16-ports switch. Switch fabric of 16-ports switch is in 32 times more than of 2-ports switch. Hardware cost ratio between ports, port controllers and switch fabric depends on many factors, especially on port controller's buffer sizes. Figure 7 represents relative hardware cost of 5 considered structures (unit is hardware cost of one channel of switch matrix). We suppose that for every structure switches exactly required numbers of ports are used. Comparison is represented in the table.

	Hardware cost of switches (buffers less than 1024)	Hardware cost of switches (buffers bigger than 1024)	Number of connectors
Structure 1	worst	best	70
Structure 2		best	86
Structure 3	best		125
Structure 4			133
Structure 5	best	worst	60

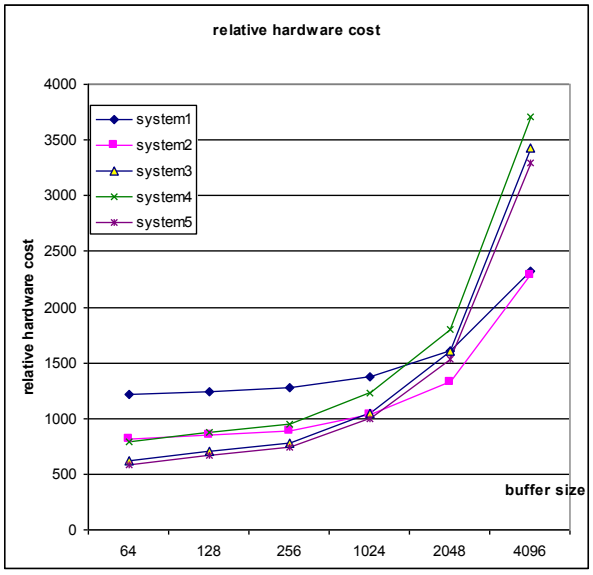


Figure 7

For systems with smaller packets daisy-chain based systems have smaller hardware cost. If packets are big, then switch based structures are better. In many applications hardware cost (weight) of cables and connectors, average and maximal length of cables, number of cables, number of connectors are very important. These parameters depend not only on interconnection graph but on real component and cable placement. In some cases switch based graphs are preferred; typically need less connectors, sometimes less cable length than for daisy-chain.

But number of cables goes via cutest for switch based system is essentially bigger than for daisy-chain. Also in view of topology for some other application daisy-chains (at least on low level) are preferred.

4 CONCLUSION

In many cases daisy-chain better corresponds system topology. Data transmission rate for daisy-chain should be in two times bigger than for switch based system for reaching same packet transmission time. Daisy-chain based systems are useful on low levels of data multiplexing in systems with small packet sizes.

UNIVERSAL SPACEWIRE INTERFACE TO/FROM VME AND TO/FROM PCI

Session: Poster Session

Short Paper

ir. G.J. Vollmuller, ing. A. Pleijsier

National Aerospace Laboratory NLR

Anthony Fokkerweg 2, 1059CM, Amsterdam / Voorsterweg 31, 8316PR, Marknesse

The Netherlands

E-mail: vollmull@nlr.nl, pleysier@nlr.nl

ABSTRACT

NLR developed in co-ordination with Satellite Services (Katwijk, Netherlands) an universal SpaceWire interface that is fully ready for production. This SpaceWire interface module is a PCI Mezzanine Format (PMC) and accommodates 3 DS SpaceWire links (Data Strobe encoding). It is connected to the PCI local bus of the PowerPC board by a high-performance 132 MB/s PCI interface.

Special features of this SpaceWire interface are:

- transmit speed up to 200 Mbit/s
- time-tagging for both incoming and outgoing packets
- wormhole routing,
- segmenting of large data structures
- priority settings for each channel
- and channel routing

This interface focuses on usage in EGSEs and SCOEs. It can be used in a PC environment (PCI bus) aswell as in an embedded PC environment (VME / compact PCI), have maximum performance and is flexible in use.

1 BLOCK DIAGRAM AND FEATURES

The block diagram of the SpaceWire PMC module is depicted in figure 1.

The central part of the interface module is the Xilinx Virtex4 FPGA that contains the ESA SpaceWire cores and the VHDL, developed by NLR, to incorporate the rest of the functionality.

The module has three SpaceWire DS links routed via FIN1102 LVDS repeaters. The Dual ported RAM is of size 512k x 36 and the module has one PCI accelerator PLX9056. The Pulse Per Second interface can be provided via a LVTTL/RS422 receiver. Power is extracted from the carrier board (3.3 Volt); the local voltages are generated on the board itself (including the start-up controller). The FRAM memory can be used for storing settings and parameters.

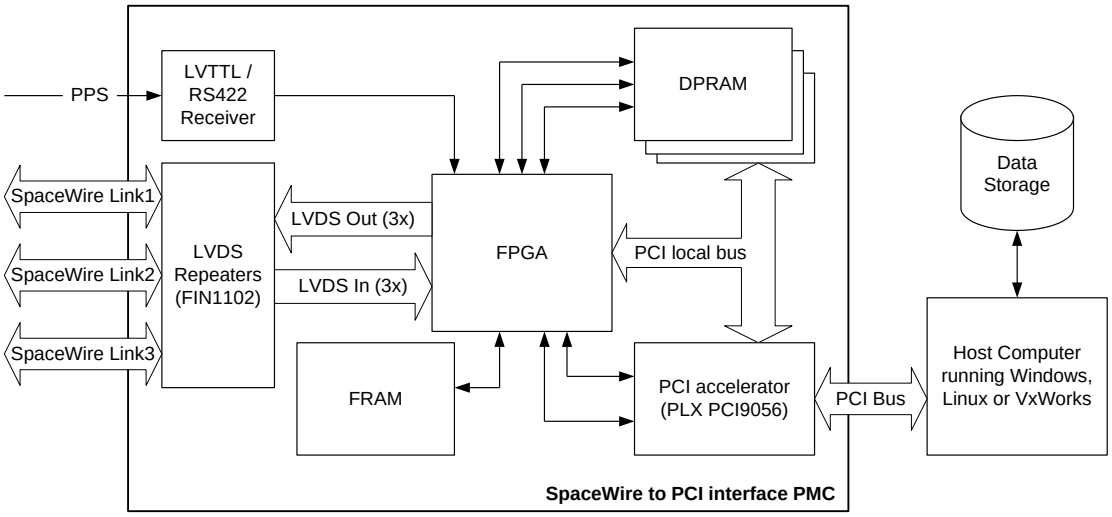


Figure 1: Block diagram of the SpaceWire to PCI interface module

The main features of this SpaceWire interface PMC module are:

- Provides three bi-directional asynchronous SpaceWire DS links via 3 ITT Cannon D-miniature connectors
- Maximum data rate of 200 Mbits/s sustained throughput on three links (measured: 260 Mbits/s)
- Provides 33/66 MHz 32 bits PCI interface
- PCI accelerator PXI9056 to facilitate PCI burst read and write cycle between Dual Ported RAM and host memory
- On-board Dual Ported RAM (512k x 36) as FIFO for temporary storage of data-to-be-send or data-received
- Pulse Per Second synchronisation (LVTTL or RS422)
- CUC timer (CCSDS Unsegmented Time Code) for time stamp of received and sent data packets
- General purpose FRAM memory (among others for storing settings)
- Temperature sensor, JTAG interface, Reset button on front panel and internal loop-back functionality for test purposes.

Note that for verification, two of these SpaceWire PMC modules are placed on a commercial Motorola PowerPC VME board; providing 6 SpaceWire links in total.

2 FPGA ARCHITECTURE

The FPGA architecture is depicted in figure 2.

The module allows communication between three SpaceWire link interfaces and a host processor. Data is exchanged between host memory and PMC using hardware initiated DMA. Data format exchange is based on so-called segments (enclosed packets or packet parts). The module allows a sustained throughput of 200 Mbits/s over three links.

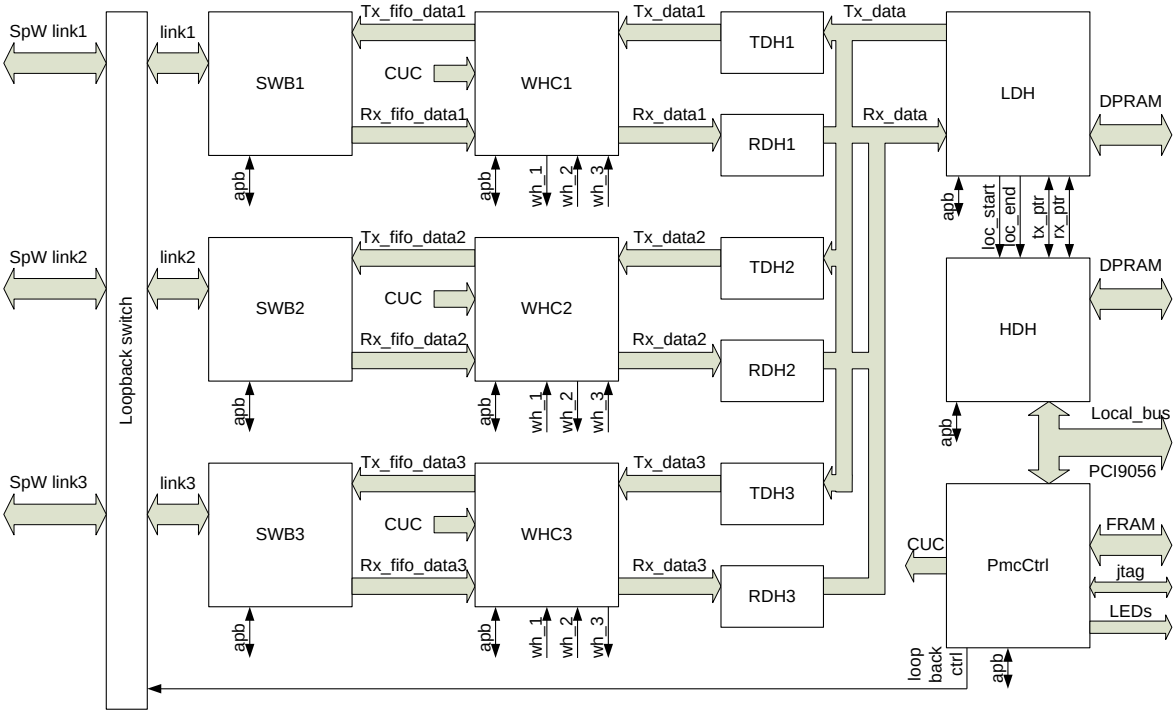


Figure 2: Block diagram of the FPGA

The Loopback switch, on the left side, offers the possibility to (re)route SpaceWire output data of a specific link to the input part of another link (or directly to its own input part). This feature can be used for PMC testing.

The SpaceWire Block (SWB) contains the ESA IP core. It allows SpaceWire packet transmission according to the SpaceWire specification ECSS-E-50-12A see [1]. The IP core also contains an AHB interface for communication to the host and uses the APB interface for internal register I/O. For the SpW PMC module the AHB (high performance bus) interface is not used for host I/O communication, but instead the FIFO interface is used to interface with the other VHDL components.

The Wormhole Controller (WHC) controls the transmitted data to a link (output packet if CUC time arrived) and controls the received data from a link (generate CUC time in segment header). It performs the route check (destination can be the host, or another link or both) and this component adds CUC times to the segment header. When the packet from the link is larger than the maximum segment size (defined by the host) the packet will be split up into segments of max_segment size. Per link interface a CUC time is maintained, based on the external CUC clock, the PPS input and host read/write CUC time capabilities. Output of a SpW packet is delayed until the requested CUC time (in the segment header) has come.

Segments to transmit are stored into DPRAM by the **Host Data Handler (HDH)**, and retrieved from DPRAM by the Local Data Handler (LDH) into the transmit FIFO of the Transmit Data Handler (TDH).

The Transmit Data Handler will read the FIFO and strip the segment header from the segment data (which is the SpaceWire packet). The output packet data (converted to 9 bits wide) is offered to the WHC, preceded by the packet status and CUC time bytes

The Receive Data Handler (RDH) controls the reception of packets. It stores the received data until the LDH is able to read this link interface. Since three channels can offer data to be stored in the DPRAM, the LDH decides (arbitrates) which channel is permitted to store data in DPRAM.

The Local Data Handler (LDH) interfaces between the link interfaces (RDH/TDH) and the DPRAM. The LDH is responsible for correct segment storage into DPRAM when SpaceWire packets from a link are received. Segment data from host to a link is retrieved from DPRAM and send to the TDH. Due to each link interface acting concurrently, the LDH arbitrates which link interface is able to send or receive DPRAM segment words.

The Host Data Handler (HDH) interfaces between PCI accelerator (PCI9056) and DPRAM in case of data exchange, and interfaces between PCI accelerator and PMC registers in case of PMC control (register read/write actions).

The PMC control (PMCCtrl) module is used for general PMC control, like controlling the Non link specific registers that are allocated in this module and handling the PMC interrupt sources. Also the CUC clock is generated in this module, using the external 33.554432 clock input signal. It also contains the FRAM I/O control.

3 API

Next to the hardware (and VHDL) development, the NLR has written the Application Programming Interface (API) for VxWorks for this SpaceWire interface module. The API contains the function calls to the PMC module. No direct (register) access to the SpW_PMC module is foreseen.

The following function calls are available:

- SpWCardOpen/Reset/Close/Status
- SpWNodeOpen/Close/Control/Status
- SpWNodeSetTime/GetTime
- SpWLinkOpen/Close/Control
- SpWLinkStart/Stop
- SpWLinkReadPacket/WritePacket/WriteStop/Status
- SpWCucControl/CucReadTime (per link)
- SpWPpsControl/LedControl/JtagStatus
- SpWRegisterRead/RegisterWrite (Enables PMC register reading/writing)
- SpWFramRead/FramWrite
- SpWDpramRead/Write
- SpWReadSegment/WriteSegment

4 REFERENCES

1. ESA-ESTEC ECSS-E-50-12A, “SpaceWire - Links, nodes, routers and networks”, Jan-2003.

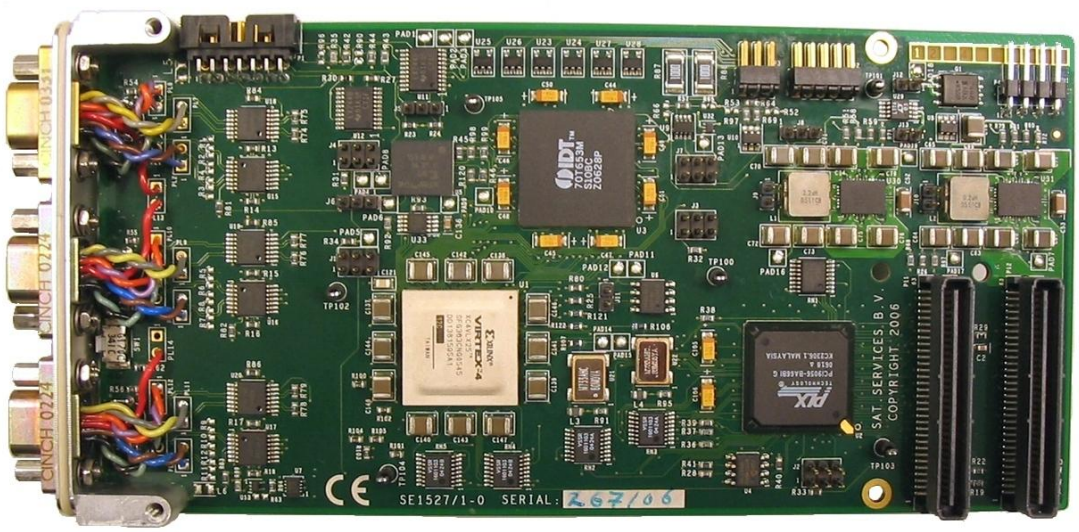


Figure 3: Picture of the SpaceWire PMC module

INCORPORATION OF THE ESA RMAP IP CORE WITHIN MARC AND THE BEPICOLOMBO RIU

Session: SpaceWire Missions and Applications (Poster)

Short Paper

P. Worsfold, A.Senior.
SEA, Building 660, Bristol Business Park, Coldharbour Lane, Bristol, BS16 1EJ, United Kingdom

Dr. W.Gasti,
European Space Agency, Postbus 299, NL-2200 AG Noordwijk, The Netherlands

E-mail:, peter.worsfold@sea.co.uk, alan.senior@sea.co.uk, wahida.gasti@esa.int,

ABSTRACT

This paper describes the incorporation of the ESA RMAP IP core within the Modular Architecture for Robust Computing (MARC) system and within the BepiColombo Remote Interface Units (RIUs).

The MARC development system implemented the IP core within Actel PRO-ASIC 3E FPGAs. Both SpaceWire Initiator and SpaceWire Target configurations have been instantiated as part of the MARC project. The BepiColombo RIUs each utilise the ESA RMAP IP core configured as SpaceWire targets within Actel RTAX-S FPGAs.

The paper summarises the implementation details of each system and the performance achieved.

1. INTRODUCTION

The ESA RMAP (Remote Memory Access Protocol) IP core is a SpaceWire interface VHDL core that includes the RMAP protocol extension to SpaceWire. It has been developed with ESA funding by the University of Dundee. SEA was selected as an Alpha user of the IP core as part of the MARC development. The RMAP IP core is now currently being integrated in to the Remote Interface Units (RIUs) for the upcoming ESA BepiColombo mission to Mercury.

This paper begins with a review of the RMAP IP core implementations within the MARC and RIU systems, then discusses the main issues encountered during its use. Full details of the RMAP IP core can be found in the University of Dundee User Manual (reference [1]).

2. MARC IMPLEMENTATION

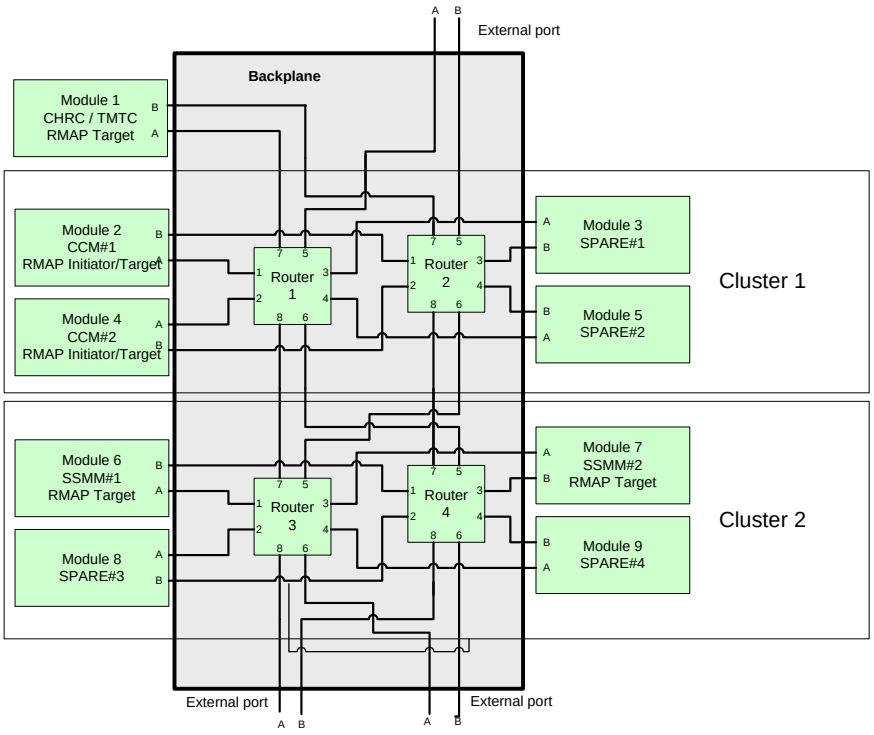
All MARC modules have two SpaceWire interfaces for redundancy and these incorporate RMAP functionality (via the ESA RMAP IP core). The RMAP protocol is defined within reference [2].

- The 2 core Computing Modules (CCMs) contain two full RMAP IP cores (both Initiator and Target enabled) controlled by a LEON2FT fault tolerant processor.

- The core Hardware Reconfiguration Controller (CHRC) contains two RMAP IP cores configured as Target only. This module provides the Failure Detection and Isolation functionality for the system.

- The 2 Solid State Mass Memory modules (SSMMs) contain two RMAP IP cores configured as Target only. These modules provide the bulk data storage for the system.

These modules are integrated onto an active backplane containing four Atmel AT7910E SpaceWire routers. The architecture is illustrated below.



The prime and redundant CCMs can access memory located on any module as an extension of its own memory using the RMAP protocol. The CCM can access software and data resources from at least 3 sources, i.e. local, the other CCM or the Solid State Mass Memory modules. The RMAP protocol provides an efficient memory access capability which also allows the system to recover from the failure of the prime CCM by storing context saving memory within the system which can be accessed by the redundant CCM.

3 MARC PERFORMANCE

The MARC implementation initialises the SpaceWire link at 10Mbit/s (as per reference [3]), then runs the links at 100Mbit/s. The IP core is configured in the TXCLK_DIV configuration driven by an external 100MHz clock for the Transmit clock and by a 25MHz clock for the system clock. In order to achieve the desired operational performance, the fastest -2 grade of FPGA was required. The large size of the RMAP IP core resulted in the largest ProASIC 3E device being used (3,000,000 gate A3PE3000).

Initially the RMAP IP core Initiator and Target functions were tested by direct connection of the modules to a University of Dundee SpaceWire Brick. Once the RMAP links were operational, the modules were plugged into the backplane, where testing of the SpaceWire links to/from the Atmel SpaceWire Routers was performed. At this stage the RMAP IP core Initiator was used to initiate the sending of packets from/to another RMAP IP core (Target) via the router network including self addressed packets. The only major issue encountered during this testing was a premature timeout of messages sent by the RMAP Initiator, however this has now been rectified by a later release of the IP core (v1.0).

4 RIU IMPLEMENTATION

The Remote Interface Unit has two SpaceWire Controller modules. Each module contains an FPGA containing two ESA RMAP IP cores. The Onboard Computer communicates with the RIU using SpaceWire RMAP packets to retreive sample data and command outputs. Both RMAP IP cores are configured as Target only (to ensure that no messages are initiated by the RIU). The design is targeted at an Actel RTAX2000S FPGA. The IP core is configured in the SYS_DIV configuration clocked by an external 20MHz oscillator.

5 RIU PERFORMANCE

The RIU initialises each SpaceWire link at 10Mbit/s (as per reference [3]), then runs the links at 2Mbit/s. The design has not yet been validated on the bench; however no issues are envisaged due to the relatively slow speed of the links (in relation to the previous MARC system).

6 IMPLEMENTATION ISSUES

6.1 CLOCK RECOVERY

One of the major issues experienced during integration was caused by repeated failures in the recovery of the receive clock from the SpaceWire Data and Strobe signal lines. This is a function of the SpaceWire CODEC contained within the RMAP IP core. The recovery is performed using a simple XOR gate. Numerous placements constraints were required to prevent the recovered clock from clocking the rising and falling edge input flip flops prior to their input setup time. The input XOR gate and input flip flops were manually placed within the FPGA and all VHDL blocks associated with the receive clock were constrained within a region to reduce skew. Actel have issued an application notice regarding this problem (reference [4]). A pre-synthesised core of this section would be a benefit for developers.

6.2 IP CORE EXTERNAL BUS

The external bus of the RMAP IP core can be sized as 1, 2, 3 or 4 bytes wide using a VHDL generic. The RMAP IP core is designed to increment addresses by 1 when using incrementing read/writes from memory, regardless of memory width. The addressable memory size therefore quadruples when using a 32 bit wide external bus instead of an 8 bit wide bus. There is no feature which allows the external address to increment as byte aligned i.e. Bits 1 and 0 are don't care on a 32-bit address. This caused integration issues when passing data pointers into the RMAP Initiator from the Atmel AT697F 32 bit wide local bus which is byte aligned. This could only be resolved with a software workaround within the AT697F code.

6.3 SEU PROTECTION

A major drawback in the use of the RMAP IP core for a space mission is that SEU protection is not provided in the current RMAP IP core model. In space applications, storage elements like SRAM are susceptible to soft (transient) errors caused by heavy ions. Space grade FPGAs, such as the Actel RTAX-S provide SEU protection for the synchronous elements in the design (flip-flops); however memory blocks are not protected. Actel memory blocks can be protected using their proprietary CoreEDAC block (described within reference [5]). The RMAP IP core therefore has to be modified to incorporate CoreEDAC blocks around the RAM elements. Small memories within the IP core are implemented as flip flops.

6.4 VHDL GENERICS

The RMAP IP core contains a large amount of VHDL Generics to configure the desired functionality. VHDL Generics are contained within both the RMAP and the SpW CODEC sections of the IP core. The user therefore has to validate their own particular configuration, which is unlikely to have been previously verified. The large amount of generics impacts the results of code coverage due to the numbers of lines which are not executed.

7. SUMMARY

The RMAP IP core is a flexible, configurable VHDL IP core which can be readily adapted to meet the varied requirements of user systems. As itemised above there are a number of enhancements which could be considered for a future release to ease its usage in future space missions. Overall the integration/testing of the RMAP IP core has been positive and no issues are foreseen with the BepiColombo RIU implementation.

8. REFERENCES

[1] SpaceNet RMAP IP core User Manual Issue 1.5, 18th May 2009

[2] ECSS-E-ST-50-11C, Draft 1.3, 01 September 2008.

[3] ECSS-E-ST-50-12C, Issue 2, 31st July 2008.

[4] Actel AC305 Application Note Implementation of the SpaceWire clock recovery Logic in RTAX-S Devices

[5] Actel AC319 Application Note Using EDAC RAM for RadTolerant RTAX-S/SL FPGAs and Axcelerator FPGAs App Note

THE APPLICATION OF SPACEWIRE IN THE DATA MANAGEMENT SYSTEM OF LSS IN WSO-UV MISSION

Session: SpaceWire Missions and Applications (Poster)

Short Paper

Chen Xiaomin, Guo Lin

Center for Space Science and Applied Research, Chinese Academy of Sciences

Cao Song, Sun Huixian

Center for Space Science and Applied Research, Chinese Academy of Sciences

E-mail: chenxm@cssar.ac.cn, guolincug@yahoo.com.cn, caosong@cssar.ac.cn, shxian@cssar.ac.cn

ABSTRACT

WSO-UV is a space astronomy project, which is under the development with joint efforts from Russia, China, Germany, Italy, Spain, the United Kingdom and a number of other countries in the world. There are three scientific instruments onboard WSO-UV: the High Resolution Echelle Spectrograph (HiRDES) lead by Germany; the Long Slit Spectrograph (LSS) lead by China; and the Slitless Spectroscopy Instrument for Surveys (ISSIS) lead by Spain. The Long Slit Spectrograph instrument is able to observe faint extra-galactic sources at the far-UV spectral range with high efficiency. SpaceWire is selected as the scientific data network onboard. The development team of LSS electronics comes from CSSAR, CAS, who are designing the data handling system based on SpaceWire at present.

This paper introduces the architecture of SpaceWire data network onboard, presents the solution for the SpaceWire interface of the data handling system of LSS, and describes the WSO-UV RDDP Transportation protocol, the QoS mechanism and design in detail.

1 INTRODUCTION

The "WSO-UV" Space Complex (SC) is intended to create a Space International Astrophysical Observatory on operational orbit. LSS is an important scientific instrument on the WSO-UV satellite. The phase A of LSS has been completed. The function of LSS is to acquire the low dispersion spectrum of the ultraviolet band at each point in the one dimension space at the same time. LSS has time tagged observation, integration observation, cycle-repeat observation, point source observation, plane source observation, simultaneous observation and other scientific observation modes. The functions of electronics unit of LSS are operation control and data handling. The electronics unit of LSS connects Science Data Management Unit (SDMU) via the SpaceWire bus of the Scientific Data Network (SDN). This paper briefly describes the SpaceWire interface hardware, protocol and design.

2 THE ARCHITECTURE OF SPACEWIRE NETWORK

LSS is connected to SDMU through the SpaceWire, as shown in figure 1[1]. SpaceWire is the primary data exchange channel between the SDMU and the scientific instruments.

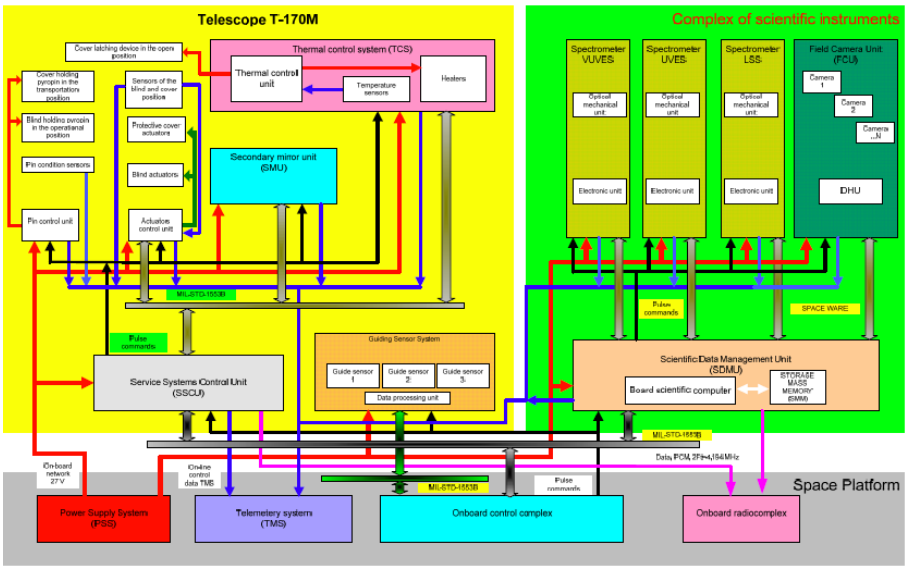


Figure 1 the architecture of SpaceWire network

Both the SDMU and all scientific instruments should be provided with two electronic control units, main and redundant (in cold redundant mode). The connection design of the SDMU and the scientific instruments should adopt the cross-strapped circuit.

3 THE SPACEWIRE INTERFACE

The SpaceWire interface unit of LSS can collect, organize and package the housekeeping data (HKTM) and observation data (STM) of Near Ultraviolet (NUV) Detector, Far Ultraviolet (FUV) Detector and Slit Viewer, and then send them to SDMU through SpaceWire, as well as receive onboard time-code and telecommands via SpaceWire.

3.1 THE CIRCUIT DESIGN OF SPACEWIRE INTERFACE

SpaceWire interface unit is composed of SpaceWire protocol controller, LVDS driver and receiver, dual-port RAM, control and interface logic circuit. The control and interface logic circuit is realized by FPGA to achieve the transport layer protocol of SpaceWire interface, and data exchange with the CPU. Figure 2 shows the SpaceWire interface block diagram.

SpaceWire interface is dual cold redundant. Each unit has three SpaceWire ports. The primary and redundant channels connect with SDMU. The remaining channel is standby or used for test. Atmel's AT7911E is applied in the design.

Synchronization is performed by SDMU periodically sending the time-code to the scientific instruments (8 times per second). The time-code conforms to ECSS-E-50-12A standard. Each eighth time-code xxx000 (binary) is associated to the onboard

second mark (1Hz pulse) at accuracy 0…+20 microseconds (TBD) without error accumulation. The Onboard Time Code (OTC) is distributed by SDMU for each scientific instrument in the data packet per second, at the time interval of 0.4 to 0.9 sec after each second mark (1 Hz pulse).

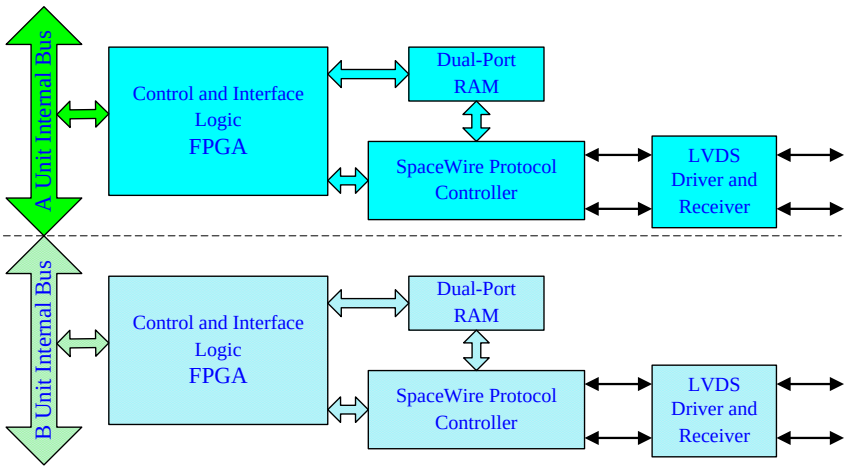


Figure 2 SpaceWire interface block diagram

All of the telecommands from SDMU to LSS, housekeeping and science data from LSS to SDMU should be transmitted as packets corresponding to the ECSS-E-70-41A in the TP envelop. The size of the telecommand, housekeeping and science data packets should be defined the same and preliminarily equal to 1024 byte (without TP envelop).

3.2 THE TRANSPORT LEVEL PROTOCAL

The SDN transport level protocol conforms to WSO Reliable Data Delivery Protocol (WSO-RDDP) [3]. It is first used in NASA Geostationary Operational Satellite R Series (GOES-R). The RDDP complies with ECSS-E-50-12A [2] and adds the following capabilities to the SpaceWire link: multiplexed logical connections, reliable data delivery; missing packet detection, out of sequence packet reordering. All WSO-RDDP packets include 8-byte header, followed by a variable length payload data, and then one byte CRC data. Figure 3 shows how the WSO-RDDP packet is encapsulated within the standard SpaceWire packet.

Destination Logical Address	Protocol ID	Source Logical Address	Packet Control
Packet Length(MSB)	Packet Length(LSB)	Channel Number	Sequence Number
Data	Data	Data	Data
Data	Data	Data	Data
Data	Packet CRC	EOP	

Figure 3 WSO-UV RDDP Packet Format

3.3 QoS MECHANISM

Each scientific instrument will send a HKTM packet to SDMU per second. To guarantee HKTM traffic reliability and timeliness, reference [4], we should complete QoS mechanism for the HKTM and STM packet transmission. The STM packets achieve assured QoS classes mentioned in SpaceWire-RT which provides a service reliably but not timely. If the guaranteed specified moment of the HKTM packet issuing within the second interval is required, each scientific instrument can be allocated its own time slot. For WSO-UV mission the QoS mechanism for HKTM provides a service which is both reliable and timely and is similar with the scheduled system in SpaceWire-RT. As HKTM and STM share the same transmission channel, we allocate different destination logic addresses to HKTM and STM, and request that logic address 32 has higher priority than logic address 48 to ensure timeliness.

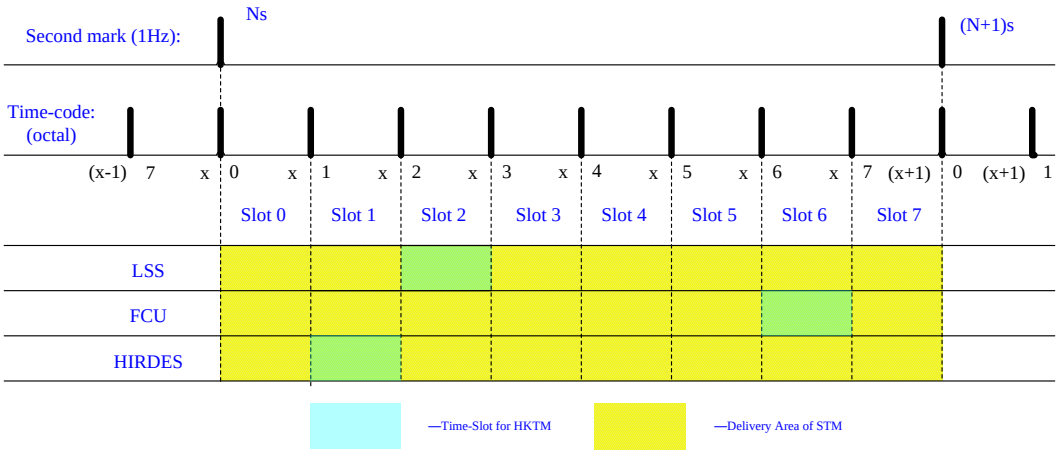


Figure 4 Packets within the second interval

As shown in Figure 4, HKTM packets are transmitted to SDMU independently and isochronously; STM packets are delivered in asynchronous mode, and are interlaced with HKTM packets.

4 CONCLUSION

The solution and designs for the SpaceWire interface, RDDP transportation protocol, the QoS mechanism are effective and meet the requirements. The above designs of LSS can also apply in the other missions in future.

5 REFERENCES

1. Lavochkin Association, “Summary of the ‘World Space Observatory (Spektr-Uf)’ Space Complex and Its Components, Initial Data on Design of the Telescope T-170m Scientific Instrumentation and the ‘WSO-UV (Spektr-Uf)’ Scientific Instrumentation Complex Equipment”, 112110-T170M-4-06, Issue 2, October 2008, P36.

2. ECSS-E-ST-50-12C, “SpaceWire: Links, Nodes, Routers and Networks”, 31 July 2008.

3. Institute of Space Research, Russian Academy of Science, "WSO-UV Reliable Data Delivery Protocol (WSO-RDDP)", Issue Draft, January 2008.

4. Steve Parkes and Albert Ferrer, "SpaceWire-RT", International SpaceWire Conference, Nara, 2008.

Components 3

ECSS TM/TC COMPONENT WITH SPACEWIRE RMAP INTERFACES

SpaceWire Components

Short Paper

Sandi Habinc, Daniel Hellström, Kristoffer Glembo

Aeroflex Gaisler, Kungsgatan 12, SE-411 19 Göteborg, Sweden

sandi@gaisler.com daniel@gaisler.com kristoffer@gaisler.com

ABSTRACT

Traditionally the implementation of telemetry encoders and telecommand decoders for space has been made in hardware, at least for the last two decades. With the availability of more processing power (e.g. LEON3-FT - 32-bit fault-tolerant SPARC V8 processor), more of the encoding and decoding tasks can be moved to software, allowing flexibility for adapting the system to on-going standardization efforts. Combining the above approach with the use of SpaceWire Remote Memory Access Protocol (RMAP) [13] results in an interesting component and software architecture.

TELEMETRY AND TELECOMMAND BACKGROUND

The return of software-based decoders in space was made in late 2009 with the launch of the European technology demonstrator satellite PROBA-2. The telecommand decoder implementation was made partially in hardware (the lower protocol layers) and partially in software (the higher protocol layers). A telemetry encoder based on similar principals has been developed, and both are being used in several on-going projects in the frame of the ESA Reference Avionics System Testbed Activity (RASTA). Both designs have been used to form the basis of a novel Field Programmable Gate Array (FPGA) TM/TC design.

ARCHITECTURE

1.1 TECHNOLOGY

The ECSS and CCSDS compliant TM/TC flight device FPGA has implemented in the anti-fuse RTAX2000S technology from Actel [18], which is radiation hard, latch-up immune, and has single-event upsets protected flip-flops. The on-chip (and off-chip) memories are protected against single-event upsets by means of BCH-based EDAC.

1.2 FUNCTIONALITY

The telemetry and telecommand concept is based on implementing the associated protocols partly in hardware and partly in software. The lower layers, such as physical layer and the channel coding sub-layer, are implemented in hardware, whereas high levels such as data link – protocol sub-layer are implemented in software. To provide the user with emergency capabilities, certain functions are implemented solely in hardware, for example pulse commands.

The TM/TC FPGA device features the following functions:

- CCSDS / ECSS compliant Telemetry encoder [2][8][9]:
  - Virtual Channels implemented in software, via a SpaceWire RMAP I/F
  - Virtual Channels implemented in hardware, via two SpaceWire RMAP I/Fs [3]
  - Reed-Solomon and Convolutional encoding in hardware [1][7]
- CCSDS / ECSS compliant Telecommand decoder [5][10]:
  - Multiple Virtual Channels implemented in software, via a SpaceWire RMAP I/F
  - One Virtual Channel implemented in hardware, with parallel pulse commands
  - Start sequence search and BCH decoding in hardware [4]

The mix between hardware and software implementation carters for a safe and sound system at the same time as flexibility is provided to support upcoming standards.

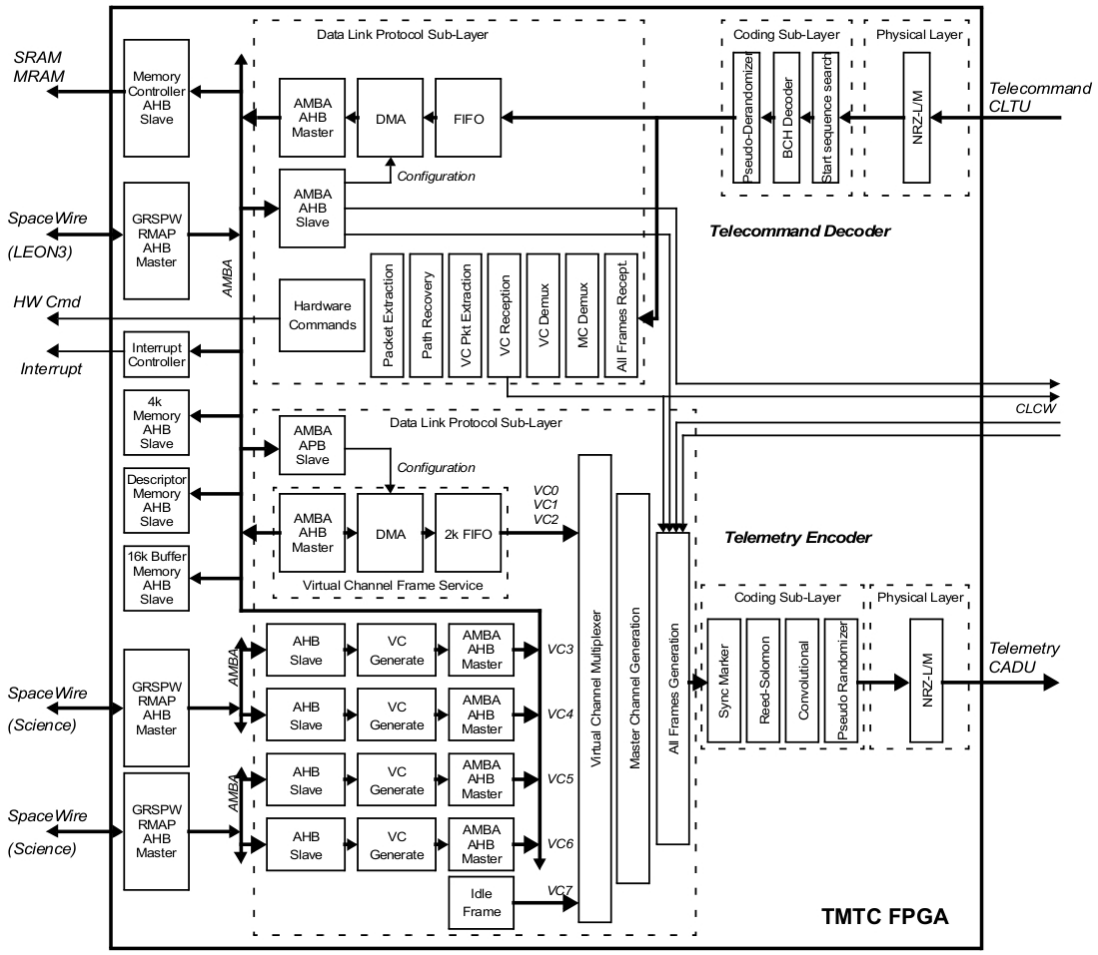


Figure: TM/TC FPGA block diagram

SPACEWIRE RMAP USAGE

1.3 INTERFACES

The main interfaces towards the TM/TC FPGA are three independent SpaceWire links [11] that implement Remote Memory Access Protocol (RMAP) [13] completely in hardware. The TM/TC FPGA implements three RMAP targets. This allows an RMAP initiator to directly access the various functions inside the TM/TC FPGA by means of

RMAP read and write commands sent over SpaceWire to the TM/TC FPGA. The various cores in the TM/TC FPGA are thus memory mapped as seen from a SpaceWire RMAP initiator. A SpaceWire initiator can be any unit capable of sending and receiving SpaceWire packets, with the RMAP protocol implemented in software.

1.4 COMMUNICATION

The novelty of this device is that the communication between the telemetry and telecommand system and on-board computer, as well as payload, is done by means of RMAP over SpaceWire links. Via RMAP read and write commands the device status can be observed and it can be controlled in a safe (verified-write command) and standardized way (ECSS standard).

For software telemetry and telecommands, complete transfer frames can be moved between the device and on-board computer. For hardware telemetry, complete Space Packets can be sent from payload to the device. The RMAP target implementation in the TM/TC FPGA requires no local processor, simplifying the design and releasing logic resources. The CCSDS / ECSS telemetry software stack [2][8] and telecommand software stack [6] are executed on an external processor and the communication is handled via RMAP.

The external processor does not need to implement RMAP in hardware. An RMAP initiator can be any device that can generate standard SpaceWire packets. The RMAP command is just a SpaceWire packet sent from the processor using its SpaceWire core. The RMAP response is just a SpaceWire packet sent from the TM/TC FPGA to the processor. A complete RMAP initiator software stack has been implemented for the RTEMS real-time operating system.

DEVELOPMENT HARDWARE

The TM/TC FPGA has been prototyped using existing development, mezzanine and accessory boards:

- GR-CPCI-AX2000 Development Board
- GR-TMTC-MEZZ Mezzanine Board
- GR-TMTC-ADAPTER Accessory Board
- GR-MRAM Mezzanine Board

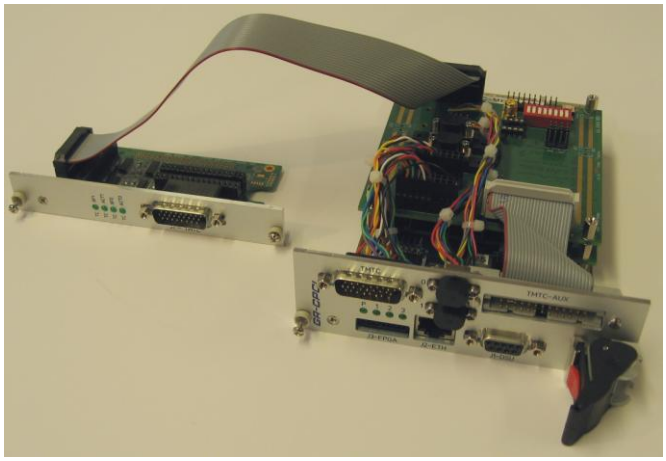


Figure: TM/TC FPGA development boards

CONCLUSIONS

The TM/TC FPGA design and development hardware has been tested and validated with the target software and has been delivered to the first customer. The various IP cores [14][15][16] used to implement the FPGA can be used in other designs, allowing custom implementations to be created. The use of RMAP over SpaceWire as the main communication channel has been proven to work efficiently.

REFERENCES

1. TM Synchronization and Channel Coding, CCSDS 131.0-B-1, www.ccsds.org
2. TM Space Data Link Protocol, CCSDS 132.0-B-1
3. Space Packet Protocol, CCSDS 133.0-B-1
4. TC Synchronization and Channel Coding, CCSDS 231.0-B-1
5. TC Space Data Link Protocol, CCSDS 232.0-B-1
6. Communications Operation Procedure-1, CCSDS 232.1-B-1
7. Telemetry synchronization and channel coding, ECSS-E-50-01A, www.ecss.nl
8. Telemetry transfer frame protocol, ECSS-E-50-03A
9. Radio frequency and modulation, ECSS-E-50-05A
10. Telecommand protocols, synchronization and channel coding, ECSS-E-50-04A
11. SpaceWire - Links, Nodes, Routers and Networks, ECSS-E-ST-50-12C
12. SpaceWire protocol identification, ECSS-E-ST-50-51C
13. SpaceWire - Remote memory access protocol, ECSS-E-ST-50-52C
14. GRLIB IP Library User's Manual, Aeroflex Gaisler, http://gaisler.com/products/grlib/grlib.pdf
15. GRLIB IP Core User's Manual, Aeroflex Gaisler, http://gaisler.com/products/grlib/grip.pdf
16. Spacecraft Data Handling IP Core User's Manual, Aeroflex Gaisler, http://gaisler.com/doc/tmtc.pdf
17. AMBA Specification, Rev 2.0, ARM IHI 0011A, Issue A, ARM Limited, www.arm.com
18. RTAX-S/SL RadTolerant FPGAs, 5172169-12/5.09, v5.4, May 2009, Actel Corporation, www.actel.com
19. CCSDS TM/TC and SpaceWire FPGA, Data Sheet and User's Manual, GR-TMTC-0002, Aeroflex Gaisler, http://gaisler.com/doc/gr-tmtc-0002.pdf

ITAR-FREE FPGA TARGETED CHARACTERISATION OF THE SPACEWIRE CEA IP

Session: SpaceWire Components

Short Paper

Cara Christophe, Pinsard Frederic, Jean-Alain Martin.
CEA Saclay DSM/IRFU/Service d'Astrophysique,
bât. 709 L'Orme des Merisiers, 91191 Gif-sur-Yvette, France.

E-mail: christophe.cara@cea.fr, frederic.pinsard@cea.fr
jean-alain.martin@cea.fr

ABSTRACT

The X- and gamma ray telescope ECLAIRs onboard the future mission for gamma ray burst studies SVOM (Space-based multi-band astronomical Variable Objects Monitor) is foreseen to operate in orbit from 2013 on. ECLAIRs will provide fast and accurate GRB triggers to other onboard telescopes, as well as to the whole GRB community, in particular ground-based follow-up telescopes. The ECLAIRs X- and gamma-ray imaging camera (CXG), used for GRB detection and localisation, is combined with a micro-channel X-ray telescope (MXT) for afterglow observations and position refinement. Sub-systems of both instruments interface with the French payload control unit (so called FCU - under CNES responsibility) by mean of SpaceWire links for PUS and CCSDS compliant message exchanges.

1. INTRODUCTION

The SVOM mission being a collaboration between the Chinese and French space agencies specific exportation rules must be fulfilled. In particularly these rules (ITAR, ...) restrict drastically components availability. This is especially critical for microprocessors and FPGAs since most of the manufacturers are in US. In this paper we discuss the implementation of the SpaceWire IP core from CEA on various ITAR-free target candidates. In particularly we present the performances and the implementation specificities for the ATF280 from ATMEL.

2. INSTRUMENT DESCRIPTION

The SVOM payload is divided into two sub-assemblies: one in under the responsibility of the Chinese space agency – so called the Chinese payload – and comprises several instruments such as the Visible Telescope (VT) the Gamma Ray Monitor (GRM) while the second in under the responsibility of the French space agency – so called the French payload – and comprises also several instruments such as the Gamma and X- Ray telescope (ECLAIRs) the Micro channel X-ray Telescope MXT). Figure 1 depicts the overall architecture of the SVOM payload. As shown each 'payload' hosts a control unit in charge of the handling of the instrument covering both command distribution, data acquisition and routing as well as health check functions. Those units interface with the spacecraft by mean of a MIL-STB-1553 bus.

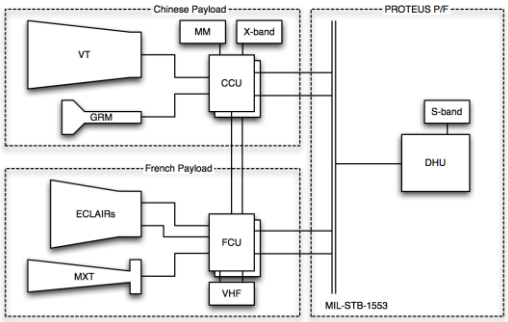


Figure 1 – SVOM block diagram

Lets now focuses on the ECLAIRs instrument whose management is under the responsibility of the CEA. Figure 2 depicts the architecture of this instrument including the camera with both optics ('coded mask'), detector plan ('pixelised' cadmium telluride) and front-end electronics (ELS) the Detector Control Unit (UGD) in charge of control of the camera and finally the Scientific Processing Unit (UTS) in charge of the spatial localisation of the gamma-ray burst either by detecting unexpected source in the field of view ('image trigger') either by detecting an excess in counting rate ('counting trigger'). Both UGD and UTS units communicate with French Control Unit (FCU) for message exchanges. The SpaceWire standard was chosen to implement these interfaces by considering the benefits of using a well-known standard. Thanks to its performances the electrical system is also well optimised since a single link can handle all kind of messages exchanged between the units.

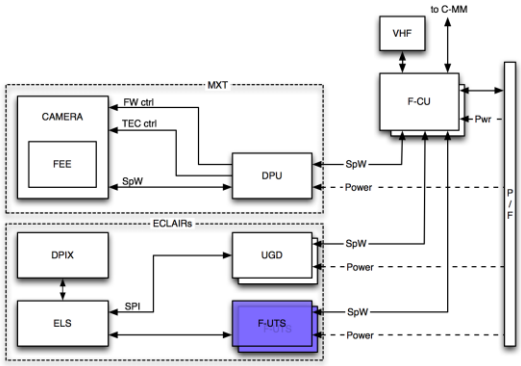


Figure 2 – French payload block diagram

3. THE CHOICE OF THE FGPA

When designing equipments with SpaceWire several options are possible: either to rely on existing implementations of the interface standard in ASIC, as provided by ATMEL or AEROFLEX companies, or either implement an IP core in an FPGA. First option minimizes development risk while the second one offers more flexibility since designers have the opportunity to host other functions of the equipment in the FPGA. It is this second option that was selected for the design of the UTS since the design relies on a LEON processor completed with FPGAs for data interfacing with the camera front end electronics and their pre-processing. Since we already had a lot of experience (HERSCHEL, SIMBOL-X, …) in the implementation of the SpaceWire

standard we felt particularly confident with this choice. Starting from that design concept of the unit the remaining open issue was the selection of the target FPGA. At this time, several years ago, we add various possibilities between XILINX, ACTEL and ATMEL products. XILINX was the best choice at least on paper thank to its performance. But rapidly we had the reject this competitor due uncontrolled risk of failure of the device. Indeed because of the high pin count CGA package the device exhibits a poor reliability under thermal cycling condition. Next choice was the RTAX family from ACTEL, and again we had to reject it not for technical reason this time but due to exportation limitations (all the parts of the family are ITAR classified). Not classified parts from ACTEL were still available (RTSX family), but limited number of cells and lack of memory were not compliant with the requirements, except eventually for the implementation of the SpaceWire. Finally the only remaining solution was the ATF280 from ATMEL manufactured in Europe and therefore free of any exportation constraint, re-enforced by classification end 2009 of the ACTEL RTSX family.

4. FPGA DESCRIPTION

The ATF280 is a radiation hardened SRAM-based reprogrammable FPGA featuring 280K equivalent ASIC gates and re-programmability. It contains 14400 logic blocks – cells- with 2 x 3 inputs and a register element, a D-type flip-flop, with programmable clock and reset polarities. Additionally it features 115 Kbits of dual-port RAM called FreeRAM^{TM}. The FreeRAM^{TM} is SEU hardened and is made of 32 x 4 dual-ported RAM blocks and dispersed throughout the array. The ATF280 has been especially designed for space application by implementing hardened cells and permanent self-integrity check mechanism (300 krad max TID and 80 MeV LET_{th}). It is available in two space-qualified packages: MCGA472 and MQFP256 packages offer respectively 324 I/Os and 166 I/Os for user application. The FGPA is available either in QML-Q and V or in ECSS quality grades.

5. IMPLEMENTATION OF THE SPACEWIRE CEA IP

The implementation of the IP is achieved by mean of the software tools provided by ATMEL. It includes a VHDL synthesiser (Precision RTL 2008a1.11 OEM_Atmel from MENTOR) a router (Atmel Figaro IDS V9.0.2) and a programming tool (SpaceProgrammer v 4.0. from ATMEL). Design verification is achieved with the VHDL simulator ModelSim (from MENTOR).
The implementation of the SpaceWire CEA IP is done ‘as it is’ and only few compiler directives shall be selected. The test set-up includes the Aerospace Development Kit (ADK) evaluation board dedicated to ATMEL space qualified processors and FPGAs with the ATF280 mezzanine board plugged-in. The ADK hosting the SpaceWire CEA IP is connected to a PCI acquisition board (PCI^{4SpW} from Skylab – Toulouse). Specific software permits the sent and the reception of data to the destination node as well as to check the link status.

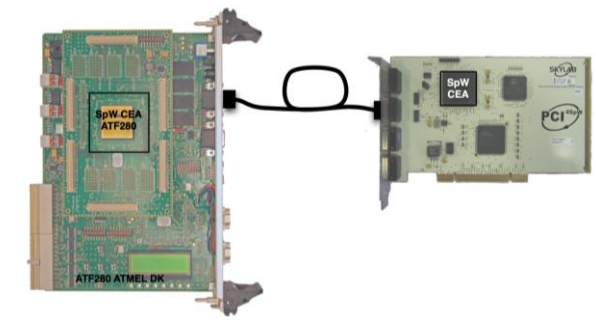


Figure 3 – Test set-up

6. RESULTS

Various configurations have been evaluated in the target FPGA. The first one consists in connecting the transmitter inputs (‘Tx’ block) to the receiver outputs (‘Rx’ block) as simply as possible (see figure 4-a). The second one consists in the implementation of a command decoder and a counter along with the IP. On reception of a ‘start / stop command’, the counter starts or stops counting and its current content is continuously sent back (see figure 4-b). This second configuration is more representative of real designs where the SpaceWire IP is supposed to interface with equipment specific functions (i.e. data processing). A third configuration was also tested (see figure 4-c) but results were not reproducible and therefore no result are reported in the present paper.

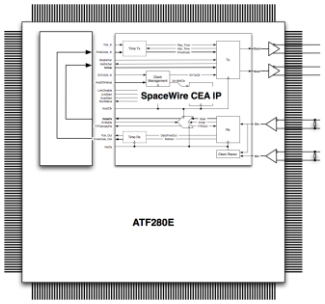
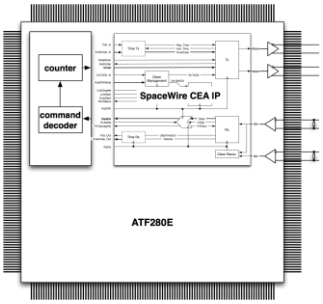
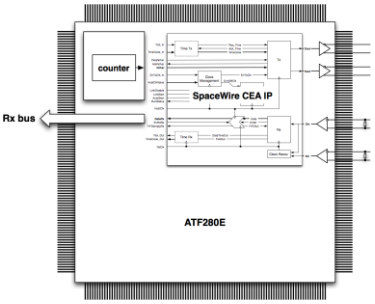


Figure 4-a: Configuration 1



-b: Configuration 2



-c: Configuration 3

Configuration 1

Frequency	10 MHz	40 MHz	100 MHz	120 MHz	140 MHz
Rx	OK	OK	OK	OK	FAIL
Frequency	10 MHz	30 MHz	40 MHz	50 MHz	100 MHz
Tx		OK			

IDS report:

```
*********************************************
Device Utilization for ATF280E-MCGA472
*********************************************
Resource                Used    Avail     Utilization
-----------------------------------------------
IOs                     9       308       2.92%
Combinational Cells     576     14400     4.00%
Sequential Cells        198     14400     1.38%
```

Configuration 2

Frequency	20 MHz	50 MHz	100 MHz	120 MHz	140 MHz	160 MHz
Link connected	OK	OK	OK	OK	OK	OK
Data received	OK	OK	OK	OK	ERRORS	NO DATA

7. CONCLUSION

These early results show promising performance of the SpaceWire CEA IP in the ATMEL ATF280 FPGA. Performance required for the ECLAIRs instrument – 10 MHz links - is already satisfied. Further tests are on-going and unpredictable operation of configuration 3 will have to be understood.

NEXT GENERATION DSP MULTI-CORE PROCESSOR WITH SPACEWIRE LINKS AS THE DEVELOPMENT OF THE "MCFlight" CHIPSET FOR THE ON-BOARD PAYLOAD DATA PROCESSING APPLICATIONS

SESSION: SPACEWIRE COMPONENTS

Short Paper

Tatiana Solokhina, Jaroslav Petrichkovich, Alexander Glushkov, Yury Alexandrov, Andrew Belyaev, Ilia Alekseev / *ELVEES R&D Center of Microelectronics, Moscow*, Yuriy Sheynin */ St.Petersburg University of Aerospace Instrumentation*

E-mail: secretary@elvees.com, sheynin@aanet.ru

ABSTRACT

The paper presents the architecture of the high performance prospective processing SoC with communication protocol SpaceWire as the Next Generation DSP Multi-core Processor for Onboard Payload Data Processing Applications.

1 INTRODUCTION

The DSP Multicore processor continues processors road map which begins from dual cores processor 0.25-um MC-24R (from the "MCFLIGHT" chipset) for signal processing and control distributed architectures with SpaceWire interconnections that is produced by the "ELVEES" company (Moscow). The structure of the chip (Fig. 1), includes the following main components:

- CPU - central processing unit based on the RISC-core and floating-point IEEE 754 standard coprocessor (FPU);
- DSP - multicore unit (from 4 up to 8 cores) for digital signal processing;
- XRAM, YRAM - the DSP memory;
- CRAM - CPU RAM;
- CDB - CPU data bus;
- MPORT - external memory port;
- DDR_PORT0, DDR_PORT1 - ports of the DDR memory;
- DMA -direct memory access controller;
- OnCD - built-in unit for the programs debugging;
- UART - Asynchronous serial port; USB Controller; I2C controller;
- AXI Switch;
- PLL - frequency multiplier based on PLL;
- Ethernet 10/100 MAC-controller;
- SWIC0, SWIC1 – standard Space Wire serial links controllers;
- GigaSWIC0, GigaSWIC1 - Giga SpaceWire serial links controllers;
- PMSC - PCI bus controller;
- VPIN - video input port; VPOUT - video output port;
- MFBSP® (Multi Functional Buffered Serial Port) – ELVEES s patented multi buffered serial port (SPI, I2S, LPORT, GPIO);
- ICTR - interrupt controller;

– IT - interval timer; WDT - watchdog Timer; RTT - real-time Clock;
– JTAG - debugging port.

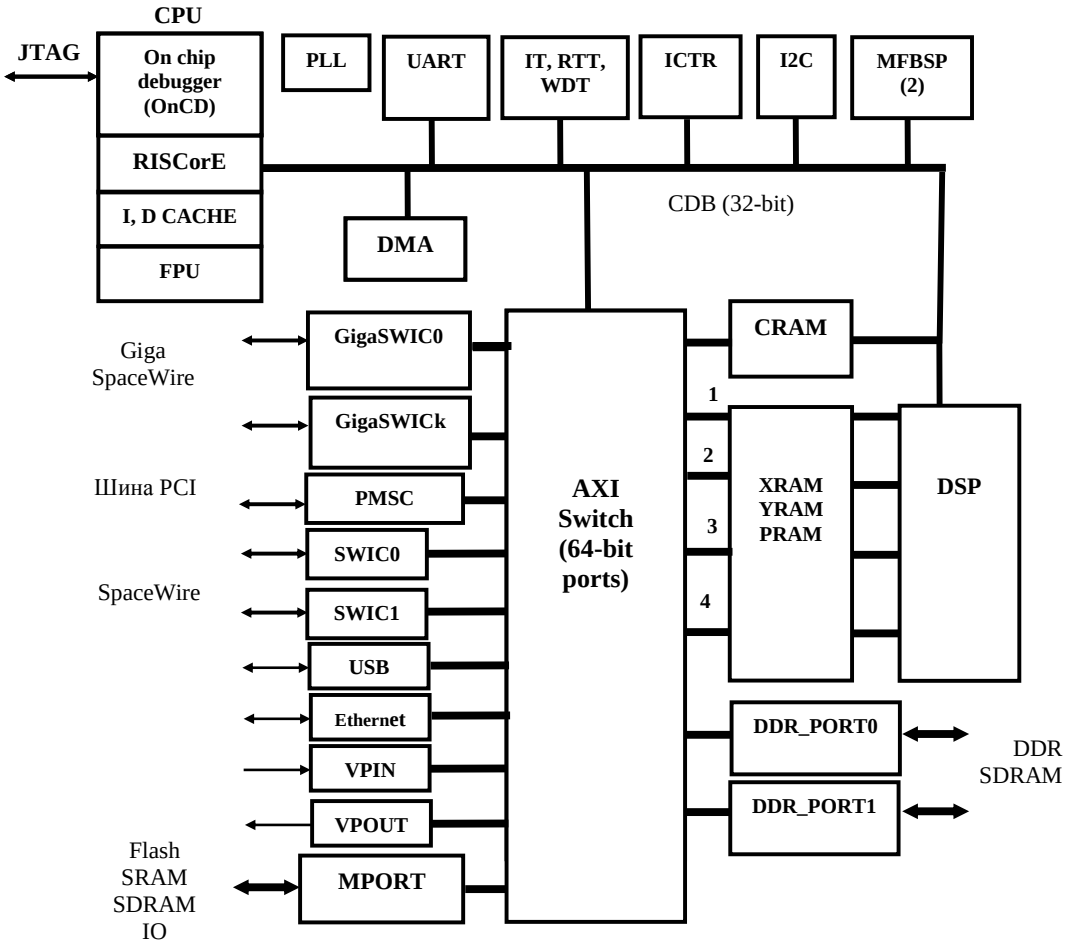


Fig. 1 Block diagram of the Multicore signal microprocessor.

Multicore signal microprocessors include the high-performance IP-cores (RISC-based CPU and DSP), embedded SpaceWire links and high throughput gigabits links, such, as sRIO, GigaSWIC® or and SpaceFibre. CPU core can be used on the base of any processor platform with an interface AXI (for example, RISCore32 (MIPS32 architecture CPU) of the MULTICORE IP-platform libraries).

As the DSP the new ELVEES 'DSP-configurations clusters from ELVEES' MULTICORE IP-core library can be used: QELcore-09™ DSP-cluster or OCTEL-core-09™ DSP-cluster. They contain 4 or 8 programmable cores (Fig.2 and Fig.3) with fixed and floating point (ELcore-™ DSP-core family, Tabl.1).

All processors in the chip operate independently of each other (each with its own program) and, therefore, represent a system-on-chip MIMD - architecture (MIMD - Multiple Instructions Multiple Data).

The most important devices in the structure of the processor are intelligent direct memory access (DMA) engines, which can provide mutual synchronizations, for example, GigaSWIC with DSP-cores.

In this case are possible both options of the MIMD-organization:

(A) CPU carries out the function of the general manager of the Executive programs, directs the DSP-cores and DMA devices, and the DSP-core is an intelligent accelerator operating on its own the program and having the

opportunity of independently initialization to implement its software. CPU has access to all processors resources.

(B) CPU and DSP - cores both have access to the chip resources. DSP have access to the entire address space of the chip, including registers, DMA-channels and peripheral blocks. With simultaneous access to the same resources the priority is given to the CPU.

2 DSP FEATURES

ELVEES DSP-cores main features:

- two 128-bit data memory accesses (X & Y) each cycle;
- vector operations, scalable 1/8/16/32/64/128 bit data formats;
- VLIW-type instruction set;
- Fixed-point and floating-point (IEEE-754) operations;
- Short (up to 7 phases) pipeline;
- AMBA AXI external bus interface;
- Non-Uniform Memory Access.

Tabl.1 ELcore-xx™ DSP-core family

DSP- core	Pipeline	SISD/ SIMD	Program memory	Data memory	Process, nm	Clock, MHz	Perform-ance, MFLOPs	Silicon proved
ELcore-14 ™	3	1	4Kx32	36Kx32	250	80	240	+
ELcore-24 ™	3	2	4Kx32	40Kx32	250	80	480	+
ELcore-26 ™	4	2	4Kx32	16Kx32	250	100	600	+
ELcore-28 ™	7	1	8Kx32	32Kx32	180	250	1500	+
ELcore-30 ™	7	1	8Kx32	32Kx32	130	300	1800	+
ELcore-09 ™	7	1	8Kx32	32Kx32	90	500	3000	-

ELVEES' DSP- cluster main features:

–AMBA AXI external bus interface;
–Non-Uniform Memory Access Architecture;
–"Floating boundary" of the data and program memory.

Tabl.2 ELcore-xx™ DSP-cluster family

DSP- cluster	DSP- core	Num-ber of cores	Program memory	Data memory (NUMA)	Process, nm	Clock, MHz	Perform-ance, MFLOPs
QELcore-28 ™	ELcore-28 ™	4	4x 8Kx32	128Kx32	180	250	6000
DELcore-30 ™	ELcore-30 ™	2	2x 8Kx32	64Kx32	130	300	3600
QELcore-09 ™	ELcore-09 ™	4	4x 8Kx32	128Kx32	90	500	12000
OCTELcore-09 ™	ELcore-09 ™	8	8x 8Kx32	256Kx32	90	500	24000

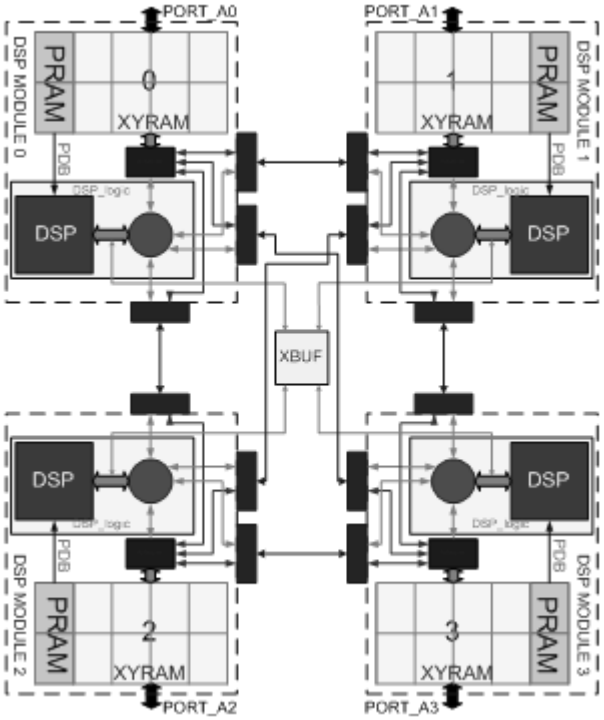


Fig.2. QELcore-09 ™ DSP-cluster block diagram

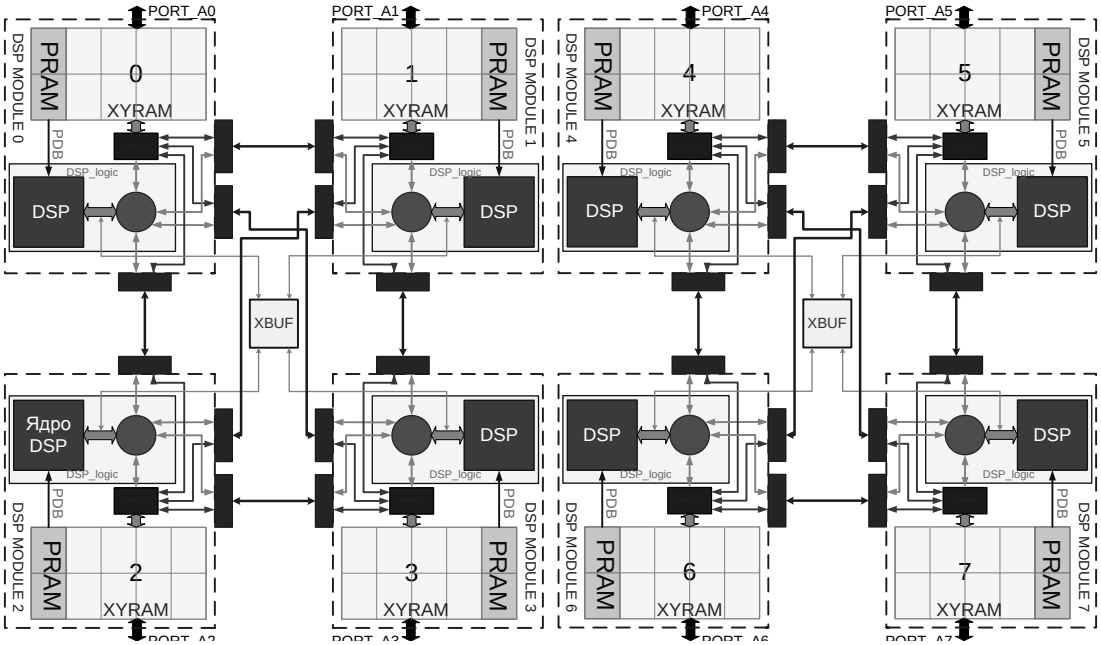


Fig.3. OCTELcore-09 ™ DSP-cluster block diagram

DSP provides the Gflops or Bops level of the performance. For example, 200 operations in 16b format or 48 floating point (IEEE754, single format) operations are performed at the same time at one clock in the OCTELcore-09™ DSP-cluster.

Therefore, necessary conditions for balancing of such a high DSP performance and data flows speed are:

A) Presence of a large number of DMA on chip devices operating with DSP;

B) Rapid channel exchanges with external devices - Giga SpaceWire serial links (GigaSWIC™).

3 GIGA SPACEWIRE SERIAL LINKS

It is important to implement the high-performance – high-rate and low latency, data transfer with low overheads and low power and area implementation costs. General purpose high-rate interconnections – Serial RapidIO, Infiniband, FibreChannel don't fit this set of requirements. The SpaceWire technology has better characteristics, though its data rate could be not enough to support the DSP-clusters performance even with parallel processing logic clustering. Higher data rates are promised by SpaceFibre (in development). Another way here could be SpaceWire evolution to Gigabit link rates. While forcing the existing SpaceWire link rates from 0,4 Gb/s in the standard (up to 0,6 – 0,8 Gb/s in some Russian and US implementations), with modification of the SpaceWire protocols low levels only, the rates of 2,5 – 5 Gb/s could be achieved in used for Multicore technologies. It uses 8b/10b coding and recoding of characters at the Symbol level. We call this modification Giga-SpaceWire, its controller core - GigaSWIC. We don't see reasonable to implement byte-level multilane links; packet level inverse multiplexing over multiple links looks more flexible and better in cost/performance space.

Along with SpaceFibre technology development finalising correspondent SpaceFibre link controller will be designed and incorporated into Multicore DSP chips along with, or instead of, the GigaSWIC cores.

The developed Multi Core DSP and scalable number (2-4) of SpaceWire/GigaSpaceWire/SpaceFibre links integrated on a System on Chip will enable the high performance on-board processing required for future missions. Integration of Multicore DSP chips into scalable multiprocessor systems is supported by direct DSP chips interconnection by SpaceWire/GigaSpaceWire links as well as routing switch chips for larger configurations with many DSP SoCs.

DSP Multi-core processor SpaceWire links are supported by drivers in Linux that run on the prototype chips.

4 CONCLUSION

The current project status: RTL code of the Next Generation DSP Multi-core Processor for Onboard Payload Data Processing Applications with GigaSWIC links, 130nm engineering samples and 180nm 5-cores DSP Multi-core Processor with SpaceWire links. The planned technology for SOC is 65 – 90 nm RAD HARD CMOS or SOI that will provide high efficiency (up to tens GFLOPs). The design and architecture must support Single – Event – Upset (SEU) fault-tolerance.

Examples of distributed computer systems Multi-Ray Satellite Relay, Earth Monitoring with small Satellite Radar onboard radar images synthesis & compression, Codec for Multi – Media Standard H264/AVC and JPEG-2000 Image compression, Coder (Convolutional, Reed-Solomon) and Decoder (Viterbi, Turbo), which are based on the "MCFLIGHT" chips, illustrate scalable reference structures for distributed signal processing and real-time control systems with SpaceWire interconnections.

Onboard Equipment and Software

A FAULT-TOLERANT SPACEWIRE COMPUTER

Session: SpaceWire Onboard Equipment and Software

Long Paper

Ran Ginosar

Electrical Engineering Dept., Technion—Israel Institute of Technology, Haifa, Isarel
and
Ramon Chips, Ltd., Haifa, Israel

E-mail: ran@ee.technion.ac.il, ran@ramon-chips.com

ABSTRACT

A study of fault-tolerant on-board computers for satellite and payload control is described. The computer comprises duplicates of six different printed-circuit boards, for a total of 12 boards. Instead of using slow redundant buses such as 1553 or dual-CANbus, the 12 boards are interconnected with multiple SpaceWire channels, four per board. A packet-switched network using One doubly-connected torus topology provides at least two mutually-exclusive paths from each board to any other board.

1 INTRODUCTION

Space computers, as well as other electronic payload assemblies, typically comprise multiple printed circuit boards, interconnected by means of a backplane, and packaged in a ruggedized box. An example is shown in Figure 1. Often, two copies of each card are included and two parallel buses are employed, enabling dual module redundancy (DMR) and increasing the resiliency and the tolerance of a single failure.

Such systems, however, suffer of two key shortcomings. First, they are susceptible to multiple failures. Second, data rates of typical buses used in space applications, such as MIL-STD-1553 and CANbus, are too low for some applications. A novel system architecture, based on SpaceWire interconnect [1], is described in this paper. It enables higher data rates and offers an enhanced fault tolerance. The conceptual architecture is shown in Figure 2. The PCBs are still arranged linearly in a box, but the interconnect is based on multiple point-to-point SpaceWire cables arranged as shown in Figure 3.

2 BUS-BASED COMPUTER ARCHITECTURE

Figure 4 depicts a simplified single-bus architecture of a space computer, comprising five cards. A DMR version is shown in Figure 5, where each card is included twice in case one copy malfunctions. Usually, one or more of the cards manage the configuration and turn other cards on or off. At times, a dedicated reconfiguration management card is assigned with this task, and is constructed with internal redundancy to mitigate single point failures.

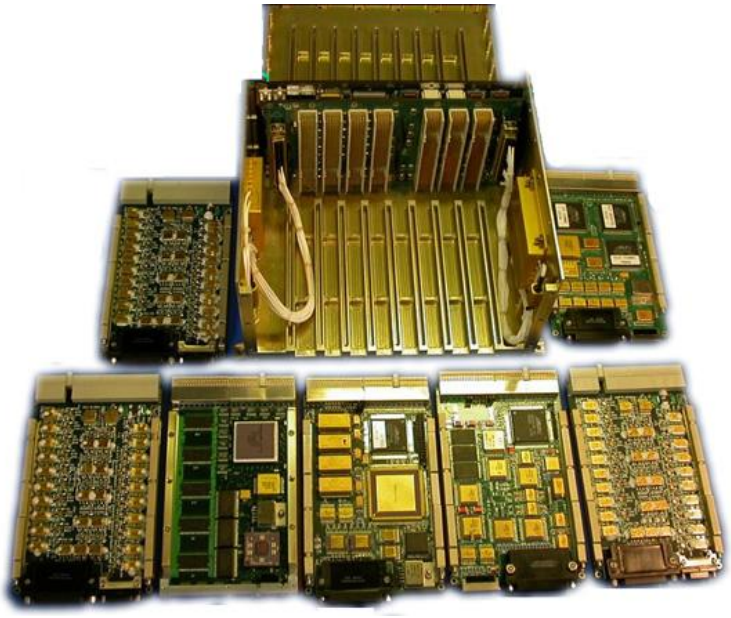


Figure 1: A typical seven-card standard format space computer based on backplane interconnect

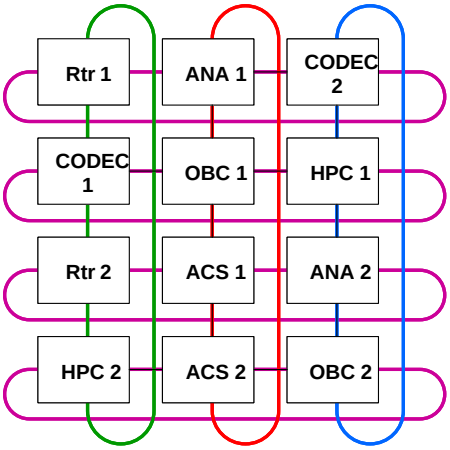


Figure 2: The SpW Computer (ANA: Analog I/O controller, CODEC: Communication port, OBC: On-board computer, HPC: High performance computer, ACS: Digital I/O controller, Rtr: Router**)**

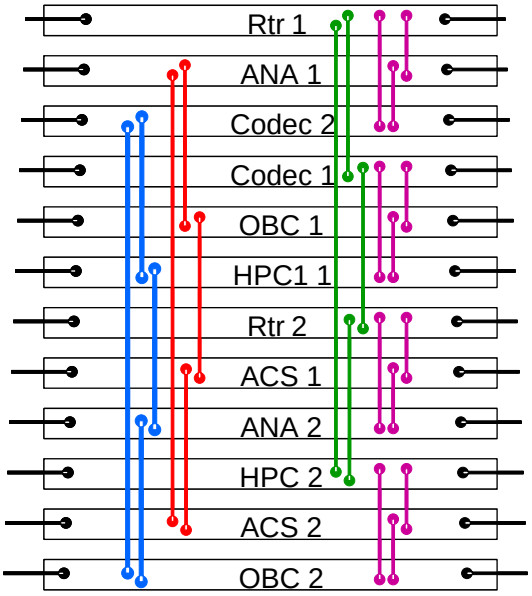


Figure 3: Cards arranged linearly in a box

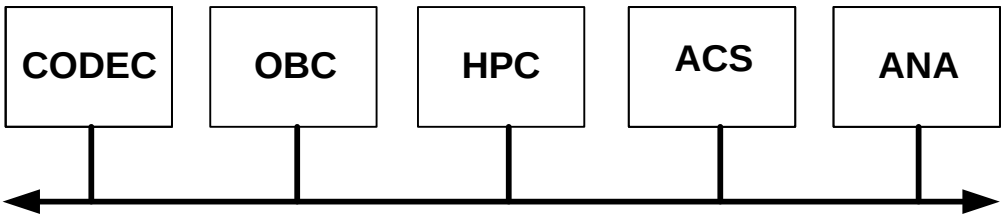


Figure 4: A space computer

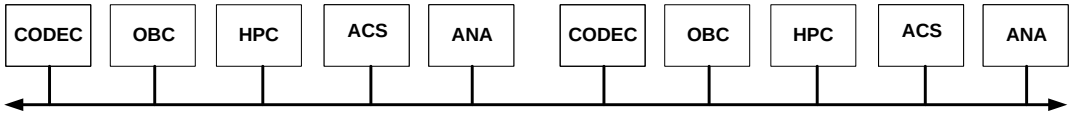


Figure 5: Duplicating all cards of the space computer for DMR

The structure of Figure 5 is still susceptible to a single failure in the bus. More reliable buses are enabled, e.g., by the MIL-STD-1553 standard which requires dual redundant buses, as in Figure 6 (1553 can also be implement with triple redundancy). Another common alternative is CANbus, implemented in duplicate similar to 1553 buses, used in order to take advantage of CANbus compatible components and to avoid export restrictions associated with 1553 components.

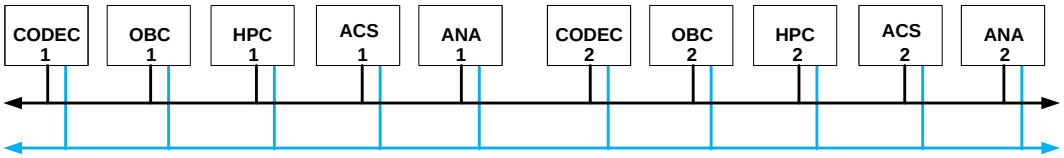


Figure 6: Dual redundant bus (e.g. as in 1553)

However, systems with dual buses are still sensitive to common failures. Indeed, a single module failure as in Figure 7 can be overcome thanks to the second copy of the failing card. However, the dual interconnect is more sensitive, because it is common to all units. If one of the two buses fails (Figure 8), the entire system is left with a single bus. Any additional failure on any of the ports, such as in Figure 9, may render the entire system unusable.

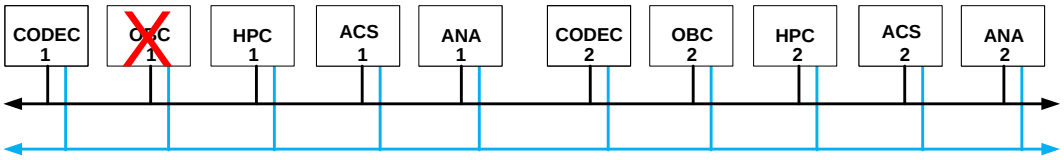


Figure 7: The DMR dual bus computer is resilient to a single module error

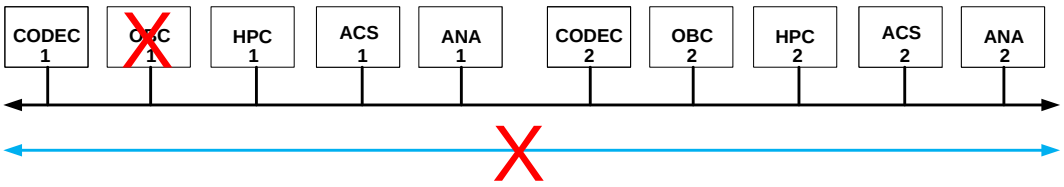


Figure 8: A single bus failure eliminates bus redundancy

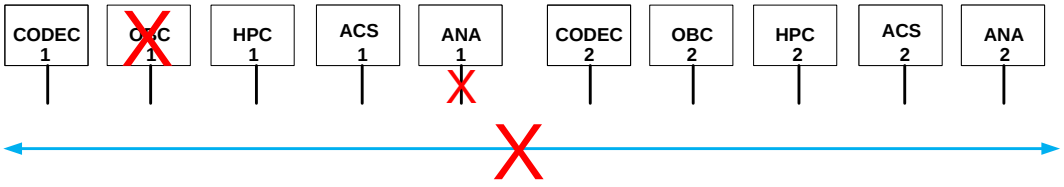


Figure 9: A bus interface failure may disable the entire computer

The other factor characterizing bus-based architectures is bandwidth limitation. Both 1553 and CANbus were designed mostly for the exchange of control data, rather than for the transfer of payload outputs or captured data. Both 1553 and CANbus are limited to 1 Mbit/s shared raw bandwidth, resulting in much lower effective sustained rates per node on the bus.

Avionics full-duplex switched Ethernet (AFDX) and various derivatives (ARINC) have been introduced as alternatives for aerospace applications. However, due to complexity and insufficient rates (about 1-2 Mbit/s at most), their use has been limited thus far. For high data rates, Ethernet requires special PHY components that complicates its application as a replacement for inter-board communications such as 1553 and CANbus.

3 THE SPW COMPUTER ARCHITECTURE

SpaceWire has been proposed as a high speed point-to-point link that enables creating redundant networks for replacing slow and fault-sensitive bus architectures [1]. The architecture of Figure 2 takes advantage of SpaceWire as the physical and data-link layers of a packet-switched network for use in multi-PCB computers. In addition to the 10 nodes of Figure 5, two extra routers are added, to increase connectivity within as well as outside the system. In packet switched networks, packets may have to pass through several intermediate nodes before arriving at its destination, and the intermediate nodes serve as routers or switches, merely passing the packet onward without affecting its contents. As common in such networks, upper layers of the network need to provide for routing, flow control, end-to-end control, reliability, retransmission, load balancing, configuration management in case of failures, and so on.

The network employs a doubly-connected torus topology (Figure 10, left). Each node is connected via four bi-directional ports, and every packet could be routed over any of the ports, creating redundancy of possible routing paths. A single node failure (Figure 10, right) results in shut-down of all four links incident upon the failing nodes; however, all other nodes remain connected.

A failure of one link causes it to disconnect without affecting the two end nodes (Figure 11, left). As above, such failure does not affect the functionality and availability of the system. In fact, many nodes and many links may malfunction and it is likely that the system will continue to perform its function, as demonstrated in Figure 11 on the right.

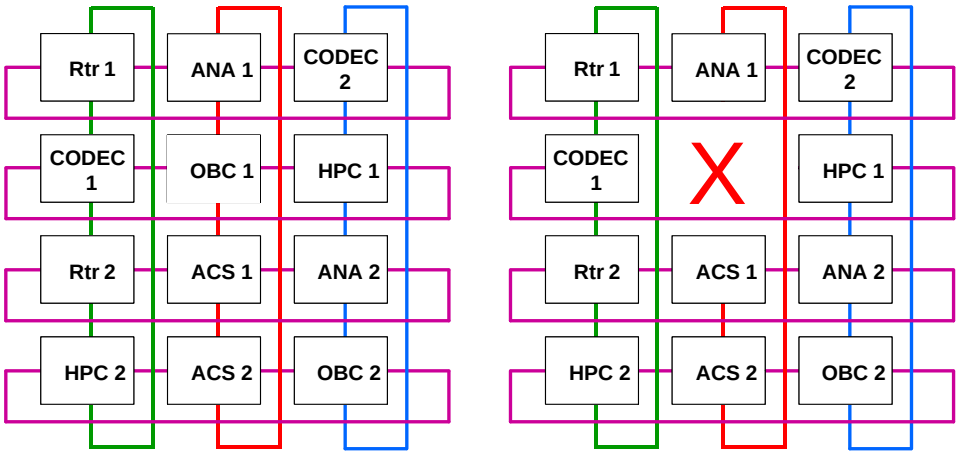


Figure 10: SpW computer (left) after one node failure (right)

Such a network-based computer systems can benefit from a multi-purpose computing-and-routing component as shown in Figure 12. Aeroflex Gaisler GR712RC is a system-on-chip (SoC) integrating two LEON3FT processor cores, large on-chip RAM and six SpaceWire ports, two of which implement the SpaceWire Remote Memory Access Protocol (RMAP, [2][3]). A typical board of the SpW computer (Figure 13) employs two copies of the SoC, one as the SpaceWire router and the other as either a main computer or as a controller managing other components on the board as well as external peripherals. The two RMAP ports of the router chip, as well as two other SpaceWire ports, make the four network connections. The remaining two ports can connect to the other processor chip (Figure 13) or to peripherals outside the SpW computer (Figure 14).

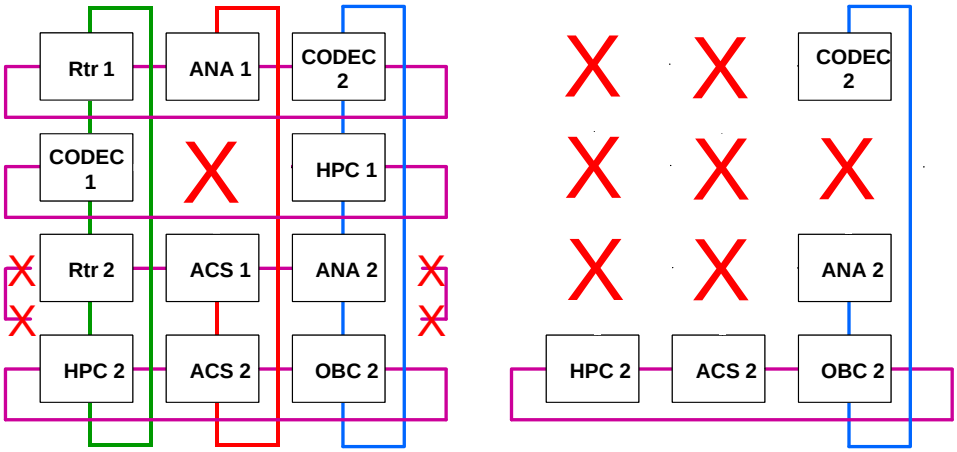


Figure 11: SpW computer after one node and one wire failures (left) and after a maximal number of failures that still allow operation (right)

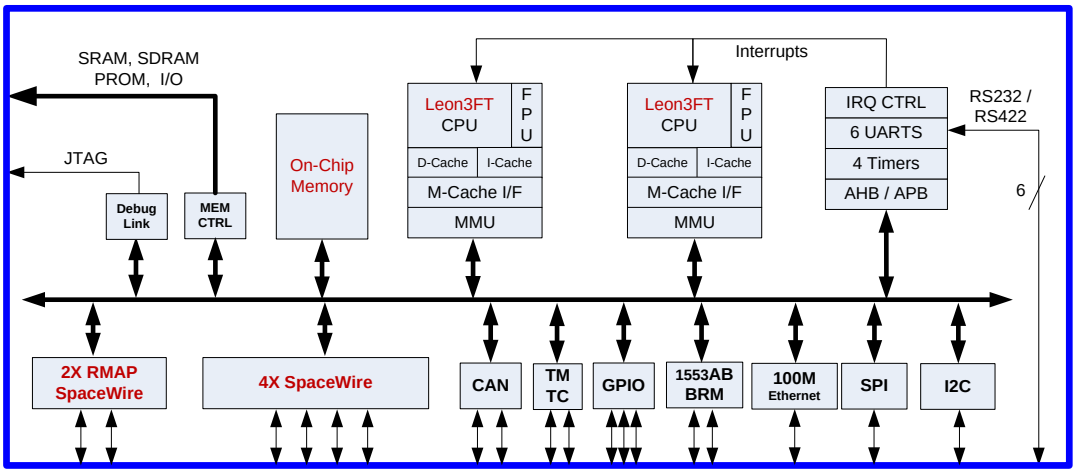


Figure 12: Aeroflex Gaisler GR712RC SoC with six Spacewire ports can serve as a router, computer, or controller

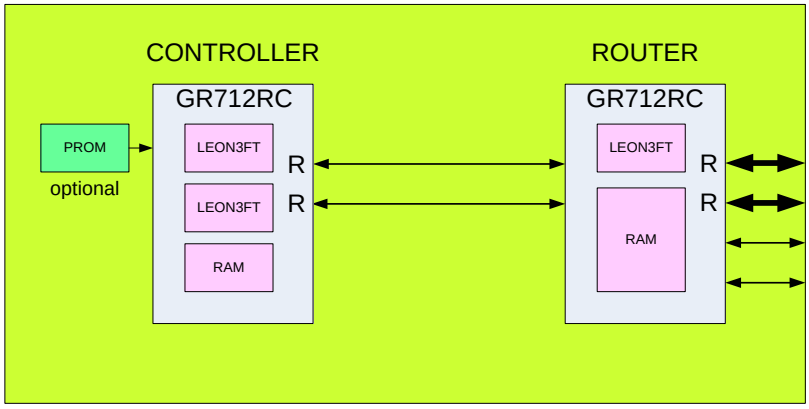


Figure 13: Aeroflex Gaisler GR712RC SoCs serving as routers receive their code over RMAP links, use the on-chip memory and do not require local PROM or RAM chips. Router Processor boards may send code to other boards over RMAP SpW links. Controller boards may receive code over RMAP links, similarly to the routers

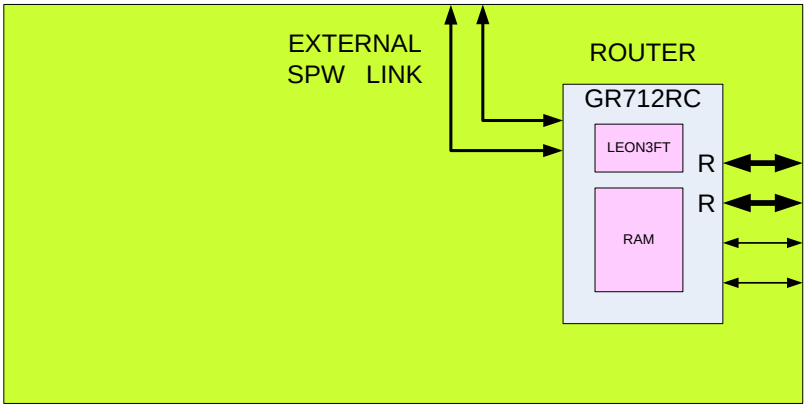


Figure 14: One or two SpW links of the router chip may be used for external links to/from outside the SpW compuetr

The advantage of RMAP ports is the ability to initialize the entire system from a single point (or two alternative points, for redundancy) as suggested in Figure 15. Thus, the network serves not only as a means of transferring data and control, but also to initialize, manage, test and repair itself.

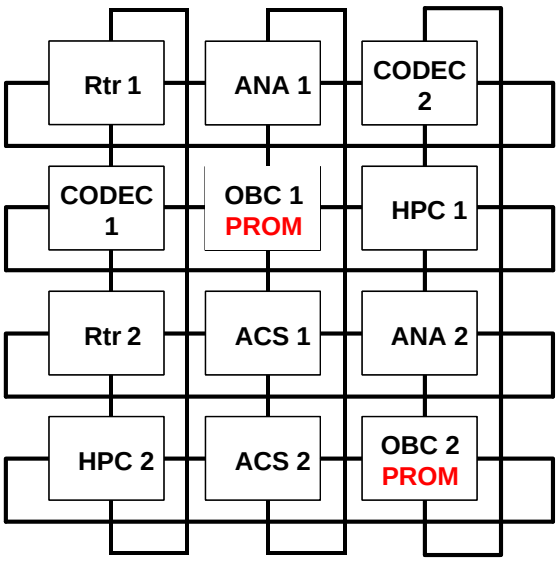


Figure 15: Only two boards in this SpW computer carry PROM code and distribute the code to all other boards (routers and controllers) via RMAP SpW links

SpaceWire links are typically designed to support up to 400 Mbit/s, providing a much higher bandwidth than any of the older buses described above. Given that in a network many (or even all) links may transfer data in parallel, the effective total bandwidth in the network described here is manifold higher than the bandwidth available in, for instance, the system of Figure 6. Even the scaled down network of Figure 11 (right) can deliver total combined bandwidth in excess of 1 Gbit/s. The actual bandwidth is typically limited by the software and by the architecture of the individual network nodes rather than by the network.

4 SUMMARY

The paper presents a SpW computer capable of high bandwidth and high level of fault tolerance. A doubly-connected torus topology offers a simple approach to the design of multi-PCB computer systems for space applications, and eliminates the need for older, slow fault-tolerant standard buses such as 1553 and CANbus.

Acknowledgements

The advice and ideas contributed by space engineering teams at the Technion, at Israel Aerospace Industries (MBT and Elta), at Ramon Chips and at Aeroflex Gaisler are greatly appreciated.

5 REFERENCES

[1] SM Parkes, J Rosello, SpaceWire- Links, nodes, routers and networks, DASIA 2001.

[2] SM Parkes, C McClements SpaceWire Remote Memory Access Protocol, DASIA, 2005

[3] S Habinc, M Isomeki, J Gaisler, The GRSPW SpaceWire Codec IP Core and Its Application, Int. SpaceWire Conference, 2007

A SPACEWIRE ACTIVE BACKPLANE SPECIFICATION FOR SPACE SYSTEMS

Session: SpaceWire Onboard Equipment and Software

Short Paper

A.Senior, P. Worsfold.

SEA, Building 660, Bristol Business Park, Coldharbour Lane, Bristol, BS16 1EJ, United Kingdom

Dr. W.Gasti

European Space Agency, Postbus 299, NL-2200 AG Noordwijk, The Netherlands

E-mail: alan.senior@sea.co.uk, peter.worsfold@sea.co.uk, wahida.gasti@esa.int

ABSTRACT

This paper introduces a SpaceWire Active Backplane (SpWAB) specification that is based on a design developed for the Modular Architecture for Robust Computing (MARC) project demonstration system [1]. The specification is based upon a modular architecture and incorporates facilities to support "Hot swapping" and "Plug and Play" system integration and test approaches.

The paper details the network and power architecture, the electrical interfaces and the potential connector design that will carry the high speed SpaceWire signals with controlled impedance between modules.

The developments required to permit realisation of the standard within a space environment are identified as well as a suggested physical configuration for the system Modules.

1 INTRODUCTION

The SpWAB specification [2] offers a communication and power distribution backplane interface that permits spacecraft units to be built from "Modules". The modules may include functions such as control processors, mass memory and Input/Output modules, potentially from different suppliers, that have a high level of capability. An example system is shown in Figure 1 which shows a reference distributed Data Handling and processing architecture [3].

The SpWAB interface to the module is comprised of redundant SpW interfaces and switched power. The module SpW interfaces are connected together via SpW routers that are part of the SpWAB, this configuration has the advantage that the network architecture is decoupled from the module design and the SpW connectivity is maintained even if individual modules have failed or have been intentionally powered off. The switched power interface permits any module to be turned off when not required, conserving power and increasing the module reliability.

The SpWAB permits a decentralised architecture system based on a SpaceWire network as a communication medium. It is anticipated that this decentralised

architecture will allow efficient resource sharing (e.g. computing, signal processing, mass memory, etc.) among spacecraft payload and platform functions.

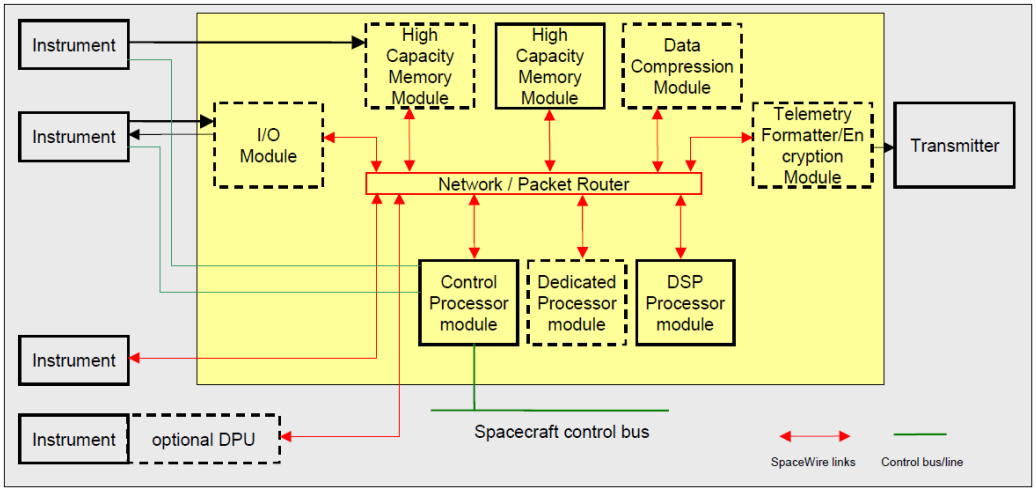


Figure 1: Distributed data handling and processing architecture

2 SpWAB Network and Power Architecture

The context of the SpWAB is shown in Figure 2. Power is supplied to the SpWAB and the Modules from nominal and redundant Power Supplies. Modules "1 to N" have identical interfaces to the backplane, while two master Reconfiguration Modules incorporate additional signal lines for power switch control.

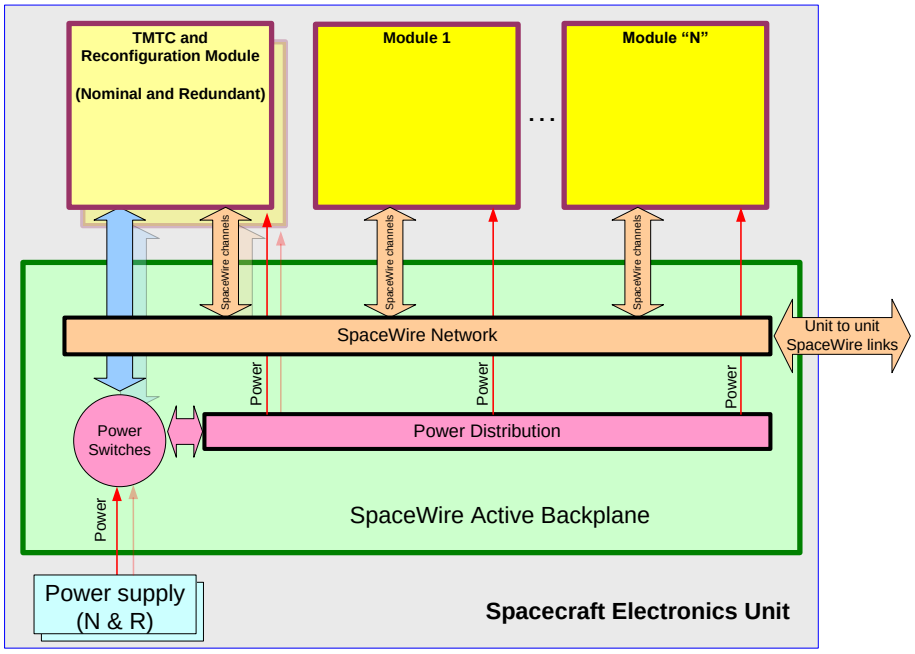


Figure 2: SpWAB context within an example Spacecraft Electronics Unit

The master Modules each incorporate a hardware based Reconfiguration Controller that handles the highest level of Failure Detection Isolation and Recovery (FDIR) within the unit. The Reconfiguration Controller performs autonomous power switching in response to an internal system watchdog timeout. In a normally operating

system the system watchdog is held off by SpW health (heartbeat) messages from one or more Modules. It is anticipated that the master Modules will incorporate either a Telemetry and Telecommand (TMTC) interface or an interface to a unit that is at a higher level in the spacecraft system FDIR hierarchy.

2.1 NETWORK ARCHITECTURE

The SpWAB network architecture is based on a Cluster (Figure 3) that incorporates two 8 port SpW routers to provide a simple network building block with built in path redundancy. The regularity of the network architecture simplifies the network discovery process needed to support a "Plug and Play" system. The Modules are identified based on the router ports that they are connected to and the Cluster number. This identification convention means that the network paths between two modules can be easily deduced. This SpWAB network can be expanded to support any number of Modules by increasing the Cluster count and it provides expansion port connections to other Spacecraft units or to Electrical Ground Support Equipment during testing.

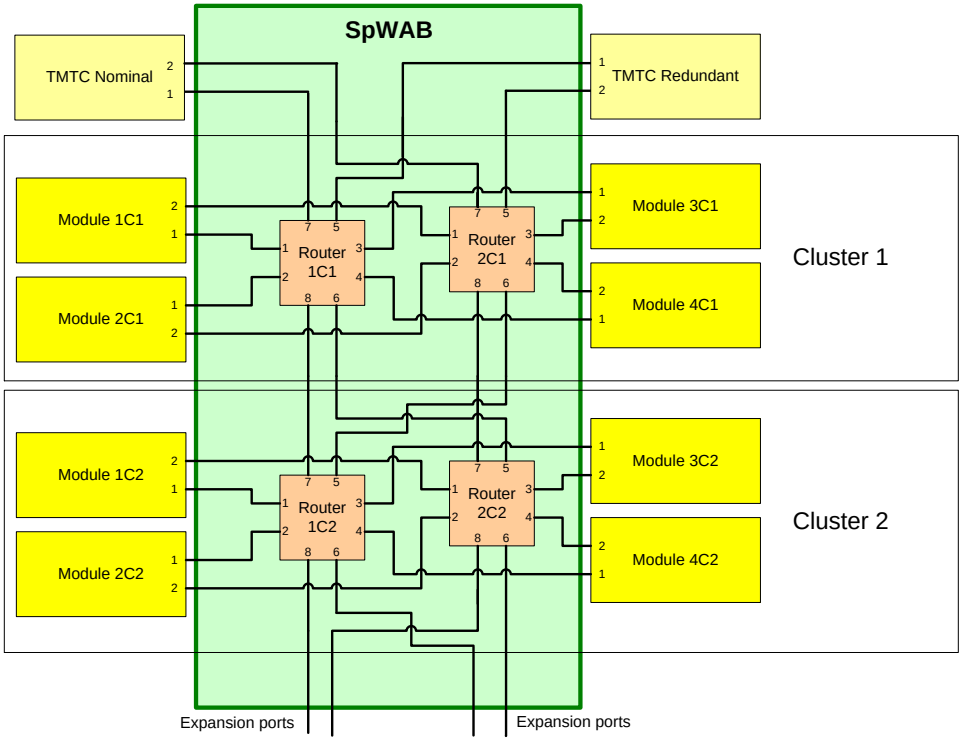


Figure 3: SpWAB network architecture

2.2 POWER ARCHITECTURE

Each Module is powered from both the Nominal and Redundant DC power rails which are combined via series/parallel diodes plus a Latching Current Limiter (LCL) for failure protection (Figure 4). The SpWAB routers are not shown in the diagram for clarity but each one is powered via a Point of Load DC-DC converter and may be switched on/off individually via LCLs.

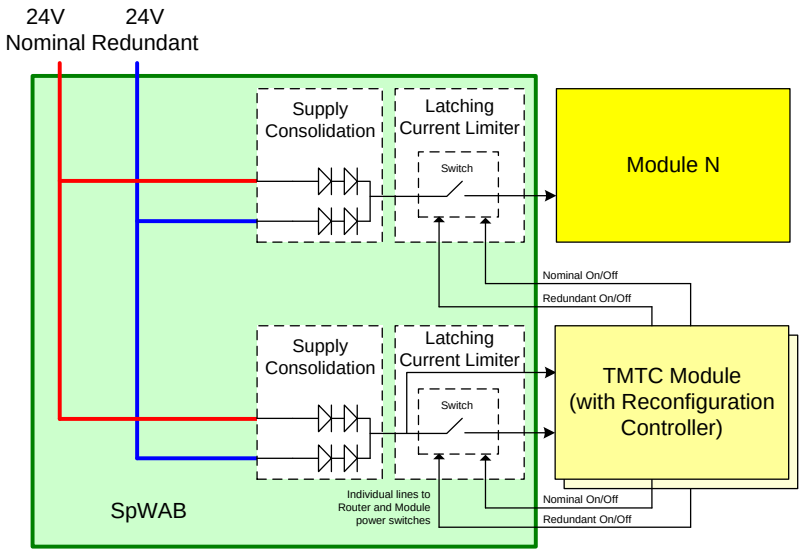


Figure 4: SpWAB Module power switching architecture

2.3 MODULE INTERFACES

Each Module is connected via 2 SpW ports to the SpWAB. No discrete signal lines are provided thus all module communication is via the SpW network. A module may have further interfaces, for example at the front panel, however these are not routed via the SpWAB.

Two SpW interfaces are required for redundancy reasons so the full functionality and performance of a module must be available if one of the SpW ports to the network fails. The module must be designed to prevent propagation of failures through the SpW interfaces by, for example, providing over-voltage protection on the internal power supply rails [4]. The Module should be compatible with the RMAP protocol for health monitoring and any basic control functions. To permit control by multiple users each RMAP control function should be accessed via atomic read/modify/write operations. To facilitate "Plug and Play" the module should map identification information into a fixed area of address space.

The module receives a single Latching Current Limiter protected rail. It is anticipated that the supplied voltage will be in the range 12V to 24V, with the lower voltage being preferred to simplify the design of low output voltage the Point of Load DC-DC converters accommodated on the Module.

3 SPWAB CONNECTORS

Currently there are no space approved backplane connectors that provide a controlled impedance interface. Options are to either, characterise an existing connector and incorporate appropriate compensation components or, to design a new connector using controlled impedance contacts. The proposed solution for the SpWAB is to develop a new pair of mating connectors in co-operation with Hypertac, the connectors being based on existing Twinax contacts and built in a Hypertac HPH form factor connector body. The Twinax contacts can operate at data rates in excess of 1Gbps and hence can handle the higher data rates that will be required for future space applications. A portion of the proposed connector contains standard contacts for

power and control signal distribution. Support for "Hot swapping" could be provided by incorporating different length pins to force removal of power upon card extraction and to ensure that the ground connection breaks last.

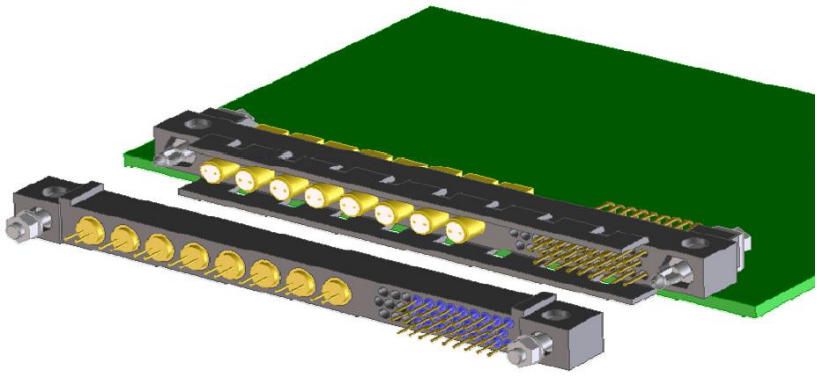


Figure 5: Proposed design of the SpWAB to Module connectors

4 MODULE PHYSICAL CHARACTERISTICS

There are no "standard" sizes for electronics circuit boards used in space applications. Spacecraft unit suppliers tend to optimise the board size for each application and produce a unit as a cube shaped box for structural, thermal and mass efficiency reasons. Clearly if Modules from different suppliers are to be integrated together within a unit then a standardised packaging scheme is required and the SpWAB specification needs to incorporate physical as well as functional and electrical interface constraints on the Module.

Eurocard board sizes are commonly used in the development of ground based electronic systems and card frames are readily available. The "Single Eurocard" size (100mm x 160mm) is considered too small to accommodate the components are required for a processor card based on the typical IC packages available in the space domain. The "Double Eurocard" (233mm x 160mm) is a more practical size but the aspect ratio leads to tall units compared to the depth. In consideration of these factors the "Extended Double Eurocard" (233mm x 220mm) is the selected card size for the SpWAB module specification.

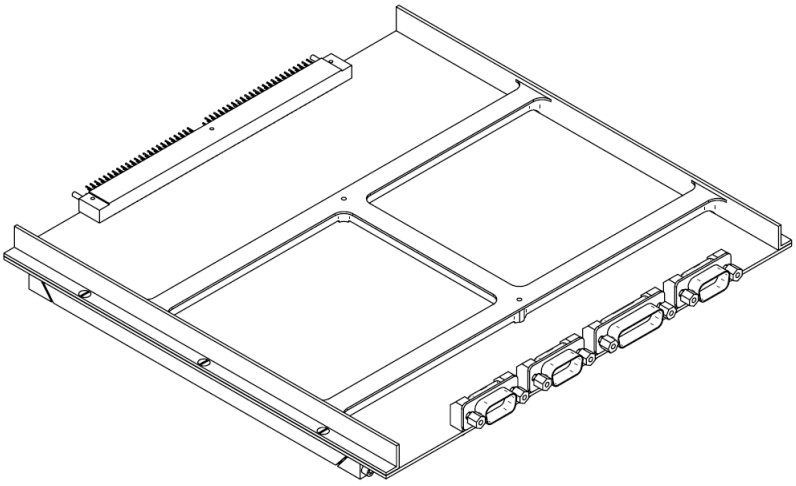


Figure 5: Example module design based on Extended Double Eurocard size

5 REQUIRED DEVELOPMENTS

To build the SpWAB as envisaged to a flight standard and permit the required components to be accommodated efficiently on the backplane three principal developments need to be undertaken:

1. Development of compact Point of Load converters incorporating over-voltage protection.

2. Develop small Latching Current Limiters with isolated redundant switch controls and possible means to monitor the current.

3. Develop a connector with a mix of controlled impedance and standard contacts that can support at least two SpW links and possibly facilities for “Hot Plugging”.

6 REFERENCES

1. A. Senior, W. Gasti, O. Emam, T. Jorden, R. Knowelden, S. Fowell, “Modular Architecture for Robust Computation”, International SpaceWire Conference 2008.

2. A. Senior, “SpaceWire Active Backplane Specification”, SEA/10/TN/9080, February 2010.

3. P. Armbruster, “Payload Data Processing Systems from requirements to implementation”, ESA Workshop on Space Data Systems System Architecture, May 2003.

4. F. Tonicello, M. Triggianese, "Points of Load Converters and distributed power architecture approach", ESA Technology Innovation Day, February 2010.

EVOLUTION OF THE MARC SPACEWIRE AND POWER DISTRIBUTION ARCHITECTURE FROM CONCEPT TO TESTED HARDWARE

Session: SpaceWire Onboard Equipment and Software

Short Paper

A.Senior, P. Worsfold.

SEA, Building 660, Bristol Business Park, Coldharbour Lane, Bristol, BS16 1EJ, United Kingdom

Dr. W.Gasti

European Space Agency, Postbus 299, NL-2200 AG Noordwijk, The Netherlands

E-mail: alan.senior@sea.co.uk, peter.worsfold@sea.co.uk, wahida.gasti@esa.int

ABSTRACT

This paper describes the evolution of the Modular Architecture for Robust Computing (MARC) demonstration system from the initial design requirements to the developed and tested system hardware. The key design decisions to achieve the modular SpaceWire and power distribution network architectures are described with the supporting rationale. The network and power architectures are based on established spacecraft redundancy concepts and provide tolerance to single point failures. The MARC hardware has been tested and the capabilities, facilities and test results are summarised.

1 INTRODUCTION

The Modular Architecture for Robust Computing (MARC) [1] demonstration hardware is a modular processing system for implementing spacecraft payload and platform avionics.

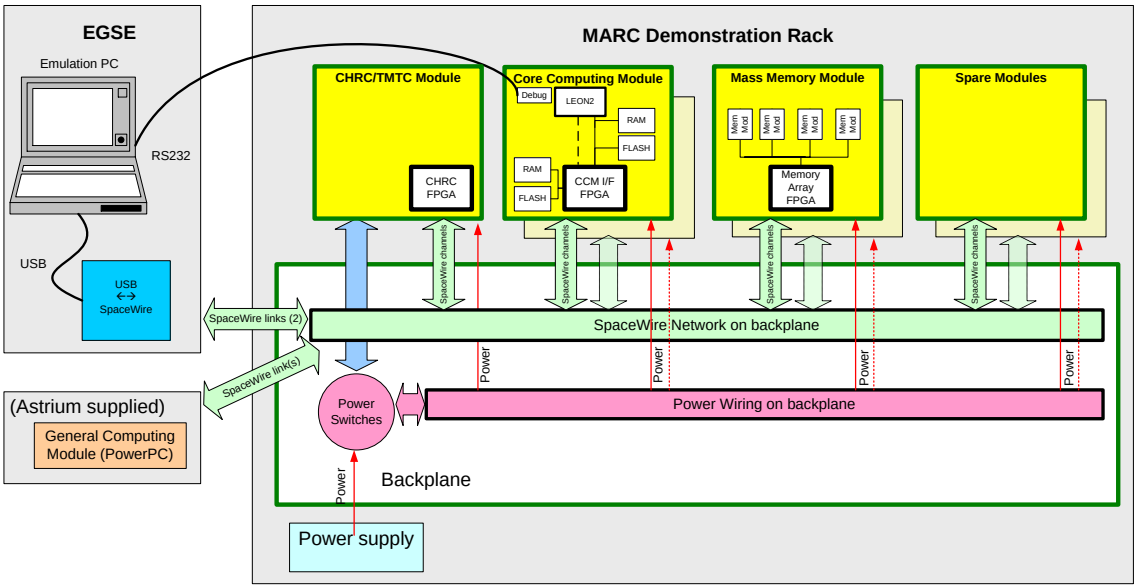


Figure 1: MARC demonstration system

MARC employs a distributed hardware architecture incorporating modules that are interconnected by a SpaceWire network; this design simplifies resource sharing (e.g. computing, communication network, memory, etc.) amongst the payload and platform functions. The main applications foreseen for this architecture include missions requiring extensive distributed fault-tolerant on-board processing capabilities, such as advanced payload data processing systems and highly autonomous space exploration systems.

A SpaceWire Active Backplane (SpWAB) [2] provides the network connections to support a set of Modules. The network can be expanded to support additional Modules or to meet the performance requirements for a particular mission. The SpW interfaces implemented in the MARC system modules use the recently released ESA RMAP IP Core [3] and the SpWAB uses the Atmel AT7910E 8 port router.

The reliable computing resource in MARC is provided by two Core Computing Modules (CCM) based on the Atmel AT697F processor. The hardware is designed to support the software architecture and the services based on the Spacecraft Onboard Interface Services (SOIS) standards.

2 MARC SpW Network Architecture

There are a variety of classical network architectures that were considered for the SpW network as illustrated in Figure 2. In this diagram the green spots could represent either a functional Module or a router. Any of these architectures could be adopted however each has advantages and disadvantages in differing applications.

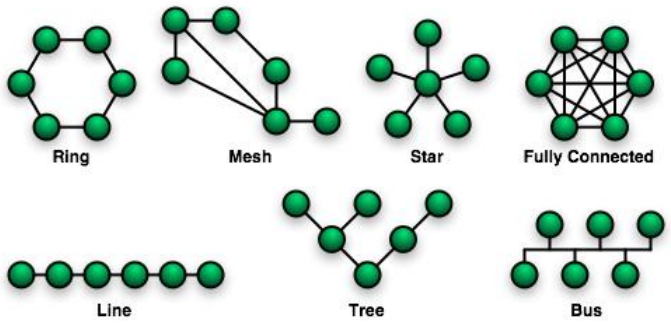


Figure 2: Classical network architectures

Single Point Failure (SPF) avoidance requires each MARC Module must connect to 2 independent routers, thus a modular network building block can be created that is naturally composed of 2 routers. In the MARC system this building block is called a "Cluster" and larger networks can be created by linking clusters (Figure 3). In principal the clusters may themselves be linked together in a variety of architectures, for example each green spot in Figure 2 could represent a Cluster. Within spacecraft practical considerations normally limit the number of Modules that need to be supported and so the simple redundant bus network architecture adopted for MARC is adequate. To support spacecraft integration and test activities spare ports are available to interface with Electrical Ground Support Equipment (EGSE). Additional spare ports are provided to connect to other external functions.

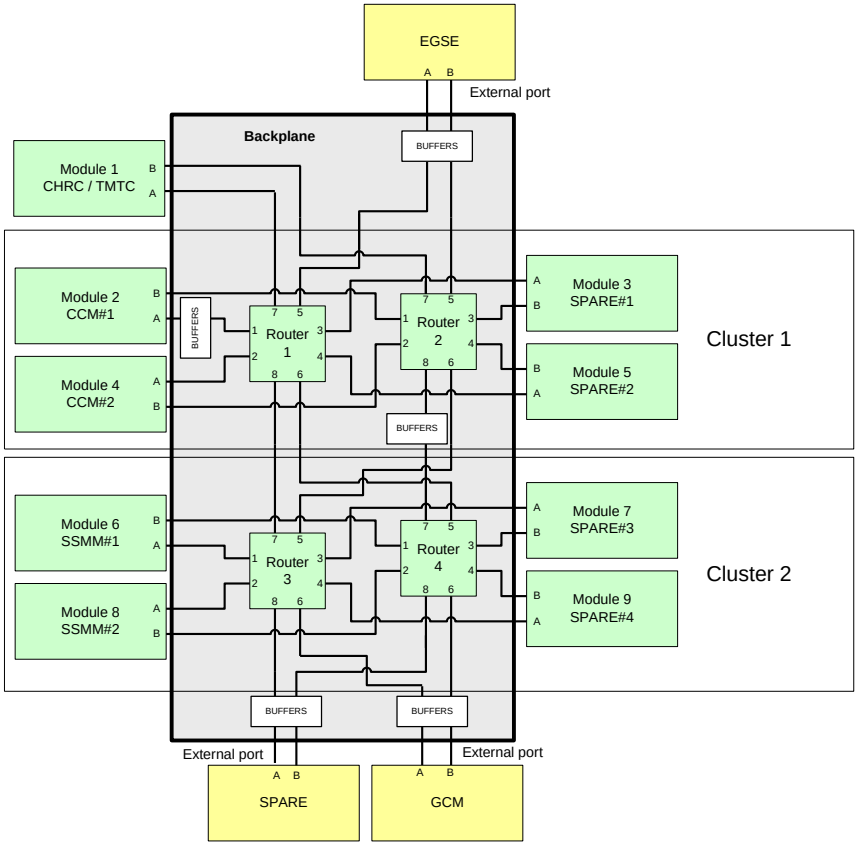


Figure 3: MARC demonstration system Cluster based SpW network architecture

3 FAILURE DETECTION ISOLATION AND RECOVERY

Failure Detection Isolation and Recovery (FDIR) autonomy is essential for space missions, in particular where recovery has to be performed without ground intervention. This is implemented by a combination of hardware and software [4]. The FDIR hardware element within MARC is the Core Hardware Reconfiguration Controller (CHRC), this is responsible for ensuring that key SpWAB routers are powered and that at least one master CCM is running. The correct functionality of the master CCM and the FDIR Manager Software running within it is indicated to the CHRC by regular SpW heartbeat messages that reset a watchdog timer in the CHRC. The objective of the CHRC FDIR action is to power a master CCM with a working communications path to the CHRC. If the CHRC fails to do this successfully it will flag the problem to the ground operator via a connection with the Telemetry and Telecommand sub-system.

4 POWER DISTRIBUTION ARCHITECTURE

It is anticipated that during a mission the MARC system will be operated in a variety of configurations that employ a subset of the available routers and Modules, this means that each router and each module must have an independent power switch. These power switches are controlled by the CHRC to permit the hardware controlled FDIR actions to be accomplished either automatically or by telecommand. To mitigate against SPFs the power distribution architecture must be comprised of Nominal and Redundant power feeds with over-voltage and over current protection [5]. The derived MARC power distribution architecture is shown in Figure 4.

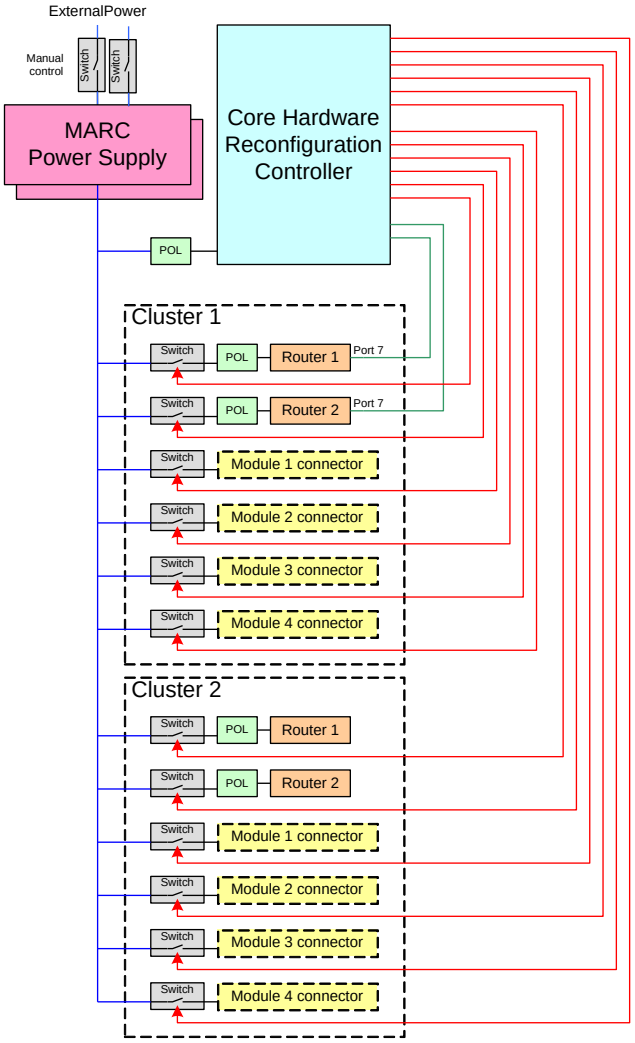


Figure 4: MARC demonstration system power distribution and control architecture

5 MARC DEMONSTRATION RACK AND TEST RESULTS

The MARC demonstration hardware (Figure 5) was completed and tested in 2009, it comprises:

- A 9U case and 6U card frame
- A SpW Active Backplane
- One Core Hardware Reconfiguration Controller Module
- Two Core Computing Modules
- Two Mass Memory Modules
- Four spare Module slots
- Rack power supplies

The ESA RMAP IP Core has been implemented in Actel ProASIC 3 devices and operates at 100Mbps (180Mbps maximum predicted by design tools).

The MARC Rack has been tested and is fully operational. The measured power dissipation figures are listed in Table 1.

Figure 5: MARC demonstration system

Module	Power	Qty	Total Power
Backplane (All links 100Mbps)	19W	1	19W
Core Computing Module	12W	2	24W
Solid State Mass Memory Module (2 x SSMBMs)	14W	2	28W
Core Hardware Reconfiguration Controller	0.5W	1	0.5W
		Total	**~72W**

Table 1: Power dissipation of the MARC demonstration system

6 REFERENCES

1. A. Senior, P. Ireland, S. Fowell, R. Ward, O. Emam, B. Green, “Modular Architecture for Robust Computing (MARC)”, International SpaceWire Conference 2007.

2. A. Senior, P. Worsfold, W. Gasti, “A SpaceWire Active Backplane Specification for Space Systems”, International SpaceWire Conference 2010.

3. C. McClements, S. Parkes, "ESA/Dundee SpaceWire RMAP Core", International SpaceWire Conference 2010.

4. A. Senior, W. Gasti, O. Emam, T. Jorden, R. Knowelden, S. Fowell, “Modular Architecture for Robust Computation”, International SpaceWire Conference 2008.

5. F. Tonicello, M. Triggianese, "Points of Load Converters and distributed power architecture approach", ESA Technology Innovation Day, February 2010.

ARCHITECTURE OF THE UNIFIED SYSTEM OF INFORMATION PROCESSING

Session: SpaceWire Onboard Equipment and Software
Short Paper

Authors 1
Eremeev P. M – the chief of department of Scientific Research Institute "Submicron",
Authors 2
Tarabarov P. A – leading engineer of Scientific Research Institute "Submicron",
Authors 3
Golovleov D.A. - leading engineer of Scientific Research Institute "Submicron".

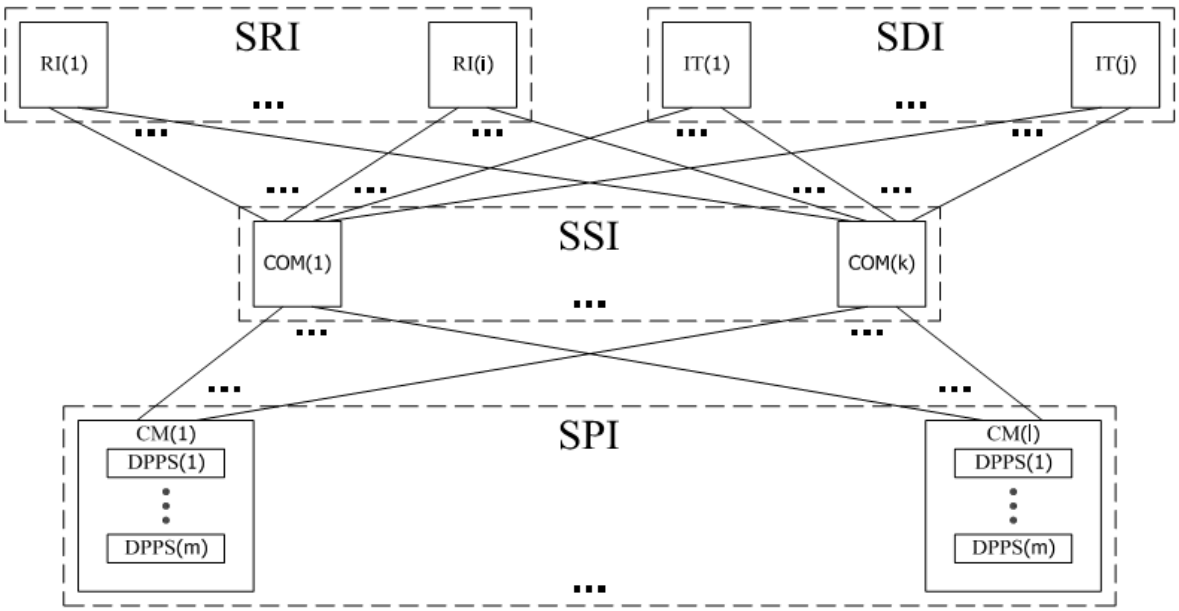
Authors 1, 2 and 3: FSUE Submicron, 124460, Moscow, Zelenograd, 4 South zone, bldg.2

E-mail: author1 epm@se.zgrad.ru, author2 sunland2007@yahoo.com, author3 mastersun1@yandex.ru

Now SpaceWire technology actively introduce in modern computing systems (CompS) - control systems (ContrS) and systems of the information processing (SIP) of onboard complexes of aircraft and spacecraft applications.

In structure of any computing system it is possibly to allocate an input-output subsystem, a subsystem of switching (information transfer) and a data processing subsystem. Also in the computing system it is separately possible to allocate a subsystem of management which is the integral component of all above-named subsystems.

Let's consider an example of structure of multimachine system of the information processing (MSIP), presented on drawing 1.



Drawing 1.

MSIP consists of 4 subsystems:

- A subsystem of reception of the information (SRI);
- A subsystem of delivery of the information (SDI);
- A subsystem of switching of the information (SSI);
- A subsystem of processing of the information (SPI).

CM - the computing module,

DPPS - the digital processor of processing of signals,

RI - the receiver of the information,

IT – the information transmitter,

COM – Commutator.

The information transfer, defining work mode and a state of receivers, transmitters, switchboards and processors is direct action of management subsystem.

Frequently the subsystem of management, a processing and information transfer subsystems are constructed on the basis of different interfaces and data transmission protocols. Design of universal architecture of multiprocessors system, in which processing and information transfer subsystems and a management subsystem are based on one interface and the protocol of high-speed consecutive data transmission, will allow to reach on qualitatively new level of multimachine systems design. It will allow:

1. To Increase the general productivity of system at the expense of use of high-speed Space Wire interface in all communication network;

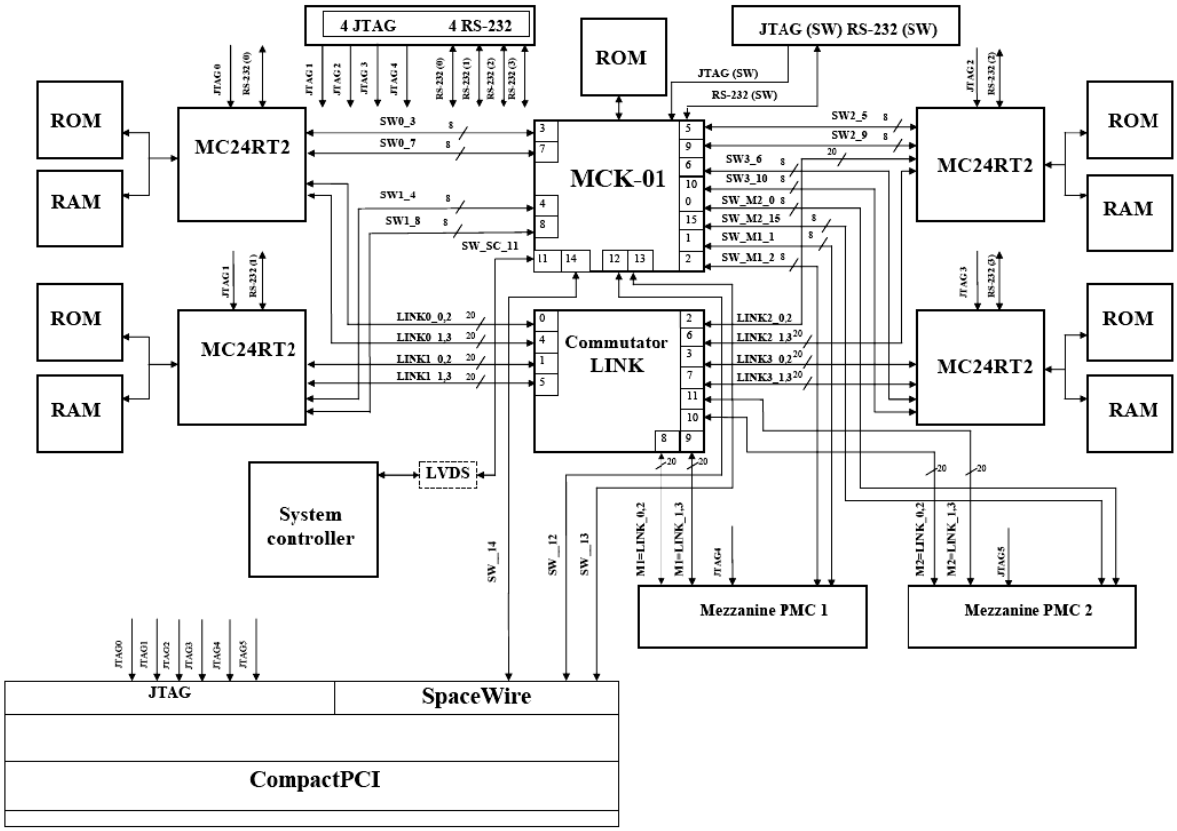
2. To Reduce time of development of hardware maintenance of system at the expense of use at all levels (modular, block, system) the same element base and identical logic and physical principles and design methods;

3. To Simplify and accelerate a development cycle of the general software of system since for transfer of the processed information and the control information is used the same protocol;

4. To Simplify testing and system adjustment.

One of current works in Scientific Research Institute "Submicron" is development of such system on the basis of SpaceWire standard. The basis of this system is the components of Scientific Production Center "Elves": signal microcontrollers of the “Multicore” series - MC24RT2, consist of two SpaceWire ports, and a integrated circuit of the switchboard of SpaceWire interface on 16 ports - MCK-01.

Let's consider block diagram CM which is used for SPI construction. The Block diagram is presented in drawing 2.



Drawing 2.

All four processors MC24RT2 in CM are connected to switchboard MCK-01 with SpaceWire. The controller exercising data control CM (System controller), also is connected to switchboard MCK-01 with SpaceWire. Two ports of the SpaceWire switchboard are connected to sockets for communication CM with SSI. Thus it is visible that switching between of processing information processors, switching of a subsystem of management, and also communication CM with SSI is carried out by one data transmission interface – SpaceWire.

The criteria of fault tolerance put in system, allow it to resist effectively to failures and faults (including to hostile faults) and to carry out detection and identification of appearing faultiness, to carry out reconfiguration at definition of the fault knot, and to carry out safe interrupt systems at impossibility to correspond to the set criteria of fault tolerance.

Thursday 24 June

Networks and Protocols 1

A SPACEWIRE EXTENSION FOR DISTRIBUTED REAL-TIME SYSTEMS

Session: SpaceWire Networks & Protocols

Long Paper

Yusuke Murata, Takuma Kogo and Nobuyuki Yamasaki

Department of Computer Science, Graduate School of Science and Technology,

Keio University

3-14-1 Hiyoshi, Kouhoku-ku, Yokohama, Kanagawa 223-8522 Japan

E-mail: murata@ny.ics.keio.ac.jp, kogo@ny.ics.keio.ac.jp, yamasaki@ny.ics.keio.ac.jp

ABSTRACT

In this paper, we propose a real-time extension scheme for SpaceWire. We designed and implemented the proposed real-time SpaceWire function on a Dependable Responsive Multithreaded Processor I (D-RMTP I) SiP (System-in-Package) for parallel/distributed real-time control, and evaluated the basic performance of the proposed SpaceWire network.

1. Introduction

In real-time systems, every real-time task has a time constraint including a deadline or a cycle. The time constraint is guaranteed by real-time scheduling algorisms. Almost all real-time scheduling algorithms for single/multi-core processors are based on pre-emption and the estimation of the worst-case execution time (WCET)[2]. Distributed real-time scheduling algorithms are being investigated by extending these algorithms. Here, pre-emption is achieved by context switches in processors. In order to employ the distributed real-time scheduling algorithms, networks are also required to be pre-emptive. Pre-emption on the networks can be achieved by overtaking prioritized packets at each node. Hence, we propose a packet overtaking scheme for SpaceWire [1].

The WCET, which is one of the most important requirements for real-time scheduling algorithms, is estimated by analyzing a program in non-distributed systems. In distributed systems, the estimation of the worst-case network latency is also required. The network latency depends on the size of a packet. Since the size of packets is not fixed on a SpaceWire network, we divide a SpaceWire packet into fixed size flits. Pre-emption of packets and the estimation of the worst-case network latency are realized by overtaking prioritized packets and the fixed size flits respectively, so that distributed real-time scheduling algorithms [6][7] especially for Responsive Link [5], which is the ISO/IEC 24740 real-time communication standard, can be applied to the SpaceWire with our proposed scheme.

We designed and implemented the proposed SpaceWire on a Responsive Multithreaded Processor [3], which is a system-on-chip for parallel/distributed real-time control, and evaluated the basic performance of the proposed SpaceWire network.

2. Real-Time Communication

Many real-time schedulers, based on classical Earliest Deadline First (EDF) and Rate Monotonic (RM) algorithms [4], have been proposed. Most real-time operating systems based on such real-time schedulers pre-empt and execute tasks in order of priority at every tick. Figure 1 shows a sample scheduling based on the EDF. The scheduling policy of the EDF is that earlier the deadline, higher the priority.

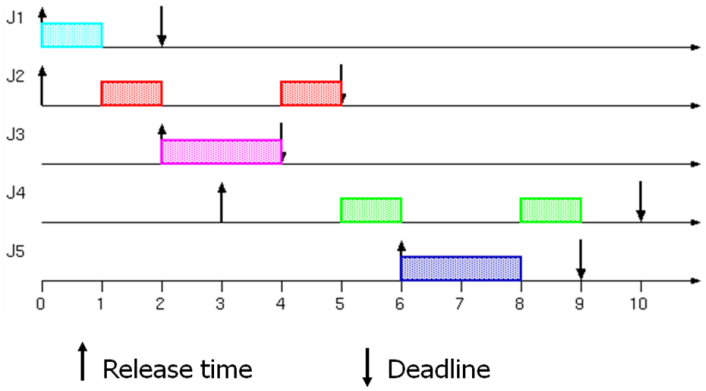


Figure 1 An EDF sample schedule

Pre-emptive processing (context switching) is required to realize real-time processing. Similarly pre-emptive communication that requires packet overtaking is needed to realize real-time communications. Therefore our goal is to realize a real-time communication architecture that can optimally do packet overtaking on SpaceWire, so that each control node can send and receive packets with suitable priority given by the real-time schedulers.

3. Packets Overtaking Scheme

We propose a packet overtaking scheme to realize pre-emption on a SpaceWire network. First we add a priority field to a routing table of SpaceWire. Figure 2 shows a routing table format for proposed real-time protocol. The routing table consists of a logical destination, a physical output, and a priority. The priority is used for packet overtaking which is realized by SpaceWire router switches with prioritized virtual channels.

Logical destination	Physical output port	priority
1	3	10
2	4	1
3	1	22
…	…	…

8bit

Figure 2 A routing table format with priority

Almost all real-time scheduling algorithms assume the known WCET. The network latency depends on the size of a packet and its blocked time. Since the size of packets is not fixed on a SpaceWire network, we divide a SpaceWire packet into fixed size flits that are control codes and data characters. We define a new control code in order to divide a packet for packet overtaking. Figure 3 shows a real-time control code format.

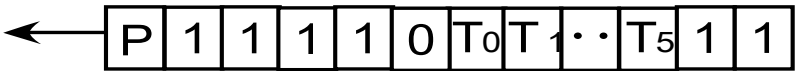


Figure 4 A real-time control code format

Figure 4 shows a packet overtaking scheme in a SpaceWire router switch. If a packet is overtaken in the midstream of a packet, the control code for the high priority packet is added to the head of the high priority packet. After transferring the high priority packet, the control code for the low priority packet is added to the head of the low priority packet, and the low priority communication restarts. The SpaceWire router switch keeps the destination address of the pre-empted packet to transfer the packet. Even if a low priority packet is divided by a high priority packet, the low priority packet that consists of discontinuous flits can be sent to the correct destination node.

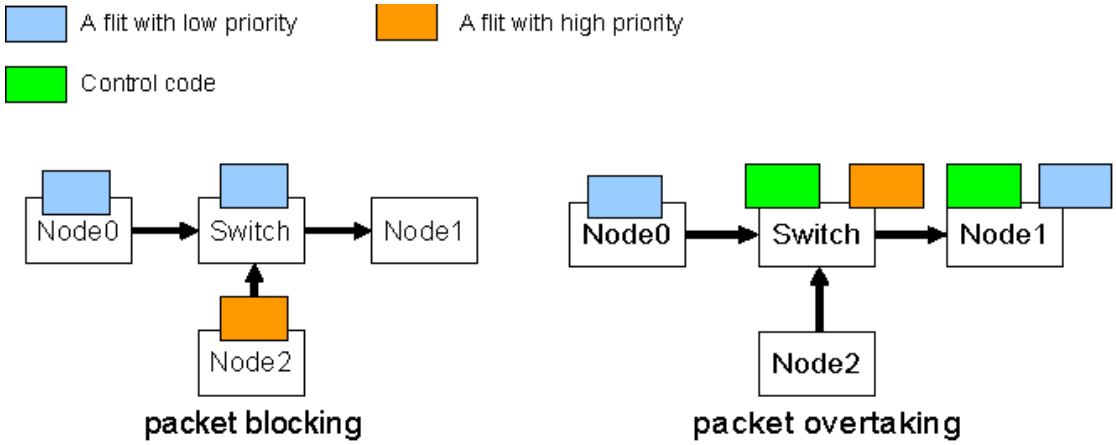


Figure 4 A packet overtaking scheme

Figure 5 shows prioritized router architecture for a real-time SpaceWire network. Since a low priority packet can be divided into a few groups of flits due to the flit-level pre-emption based on priority in the real-time SpaceWire network, a mechanism that merges the divided packet that consists of the groups of flits into the original packet is required. In case of flit-level pre-emption based on priority, when a router restarts sending a low priority packet that was pre-empted by a higher priority packet to the corresponding virtual channel of the next router, the virtual channel should be specified correctly. In order to solve this problem while keeping compatibility of SpaceWire networks, our real-time SpaceWire scheme uses a control code of the SpaceWire protocol for flit overtaking. The proposed real-time control code indicates the correspondence relation between the virtual channel of the current router and the virtual channel of the next router. When a pre-emption occurs at a router, the router

switch generates a real-time control code for the high priority packet, and the router switch sends the real-time control code to the next router. Then the router switch sends the flits of the high priority packet. At the same time, the router controller generates a real-time control code for the low priority packet that also indicates the virtual channel of the low priority packet of the next router. The real-time control code for the high priority packet is sent to the next router and switches the virtual channel of the next router correctly. Specifically the high priority packet is buffered to another virtual channel for the high priority packet at the next router. When the router switch finishes sending the high priority packet, the router restarts to send the low priority packet. At this time, the real-time control code for the low priority packet that indicates a restart of the low priority packet is sent to the next router to switch the virtual channel correctly. Communication of the low priority packet restarts, so that the low priority packet is sent and buffered to the reserved virtual channel for the low priority packet at the next router.

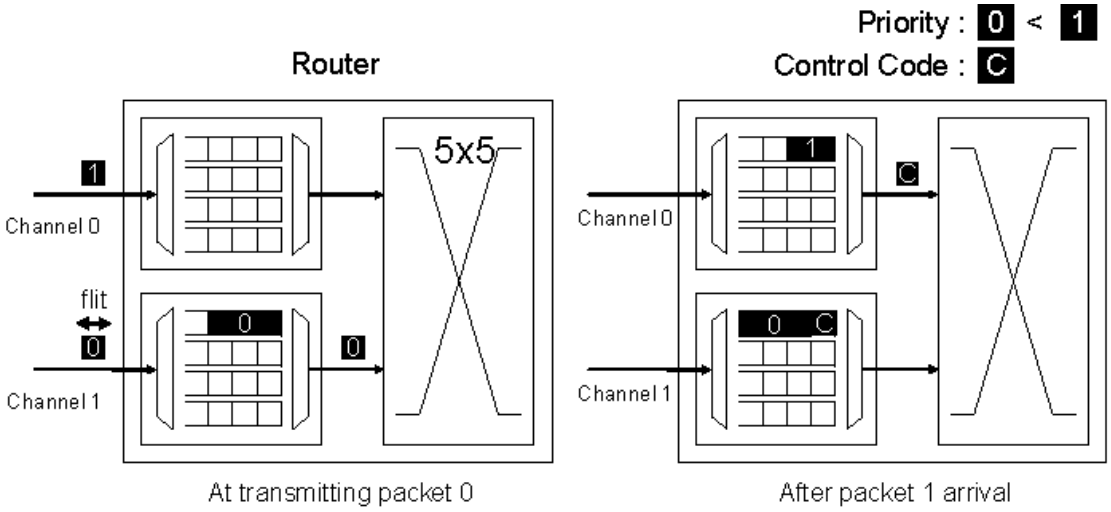


Figure 5 A prioritized router

4. Implementation

We have implemented the proposed SpaceWire router switch in the Spartan3e FPGA on the D-RMTP I (Dependable Responsive Multithreaded Processor I) SiP (System-in-Package) as shown in Figure 6. The D-RMTP I SiP, which size is 3x3cm, integrates the D-RMTP I, four DDR SDRAMs, two flash memory chips, Ethernet phy, an FPGA (Spartan3e), etc. The D-RMTP I, which size is 10 x 10mm, is a SoC (System-on-Chip) for distributed real-time control, which integrates a real-time processing core (RMT PU: 8-way prioritized SMT with 2D vector units), SRAM, DDR SDRAM IF, PCI-X, SPI, IEEE1394, Ethernet, PWMs, encoders, UART, etc into an ASIC chip as shown in Figure7. All D-RMTP I functions except the SpaceWire router switch are integrated into the D-RMTP I chip. The FPGA (Spartan3e XC3S500E) is exclusively used for the SpaceWire router switch, so that the protocol of the real-time SpaceWire can be changed easily.

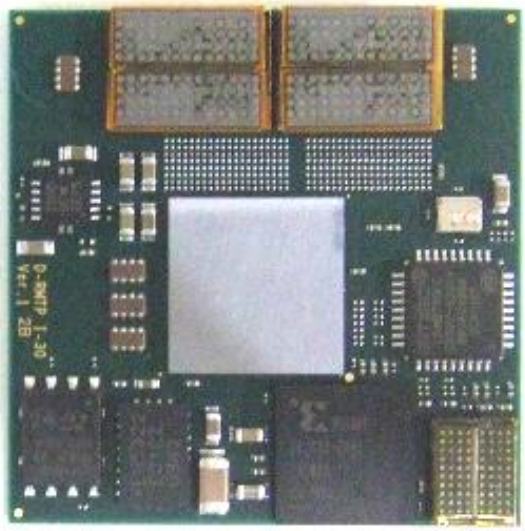


Figure 6 Photo of D-RMTP I SiP

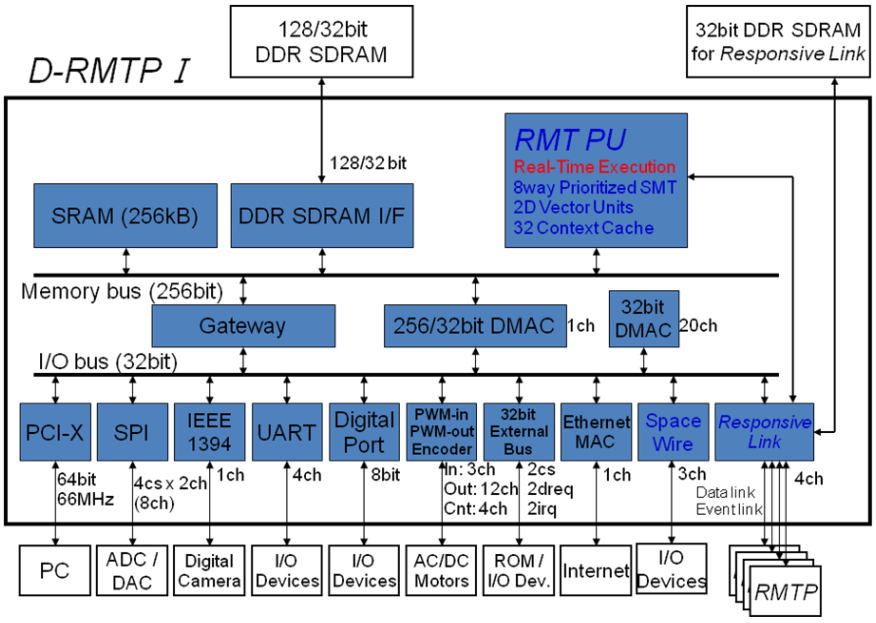


Figure 7 Block diagram of D-RMTP I SiP

5. Evaluation

We implemented the SpaceWire router switch written by Verilog HDL on the FPGA. We evaluated the SpaceWire network by RTL simulation using NC-Verilog. Figure 8 shows a network topology for the evaluation. Each node generates 64-byte packets, which destination addresses are changed at random, under random uniform traffic. The average latency of packets and the maximum latency of packets were measured, while the network utilization was changed.

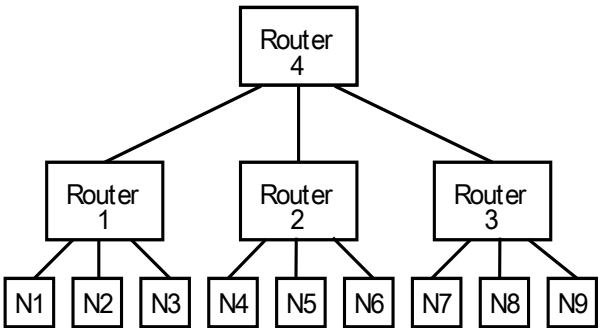


Figure 8 Network topology

Figure 9 shows a basic preliminary performance of the proposed SpaceWire network, which shows a network latency without priority. While the network utilization becomes higher, the average latency of a packet does not increase so much, but the maximum latency of a packet increases.

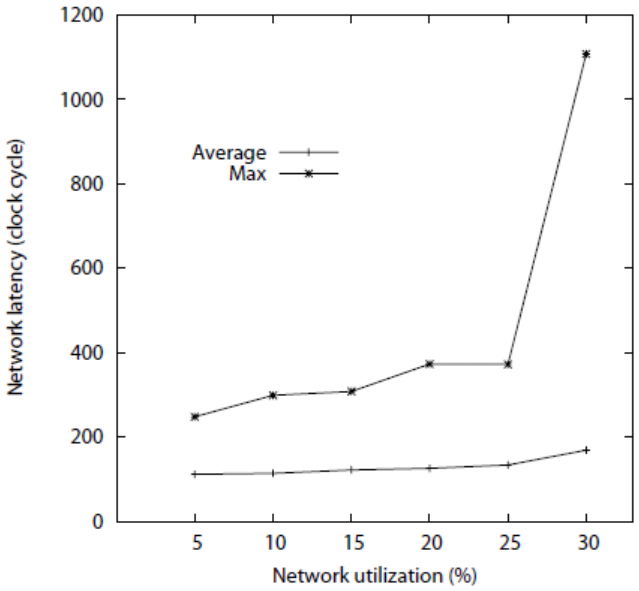


Figure 9 Network latency (base line)

6. Conclusion and future work

We proposed a real-time extension scheme for SpaceWire. We designed and implemented a packet overtaking function for real-time SpaceWiare networks. We designed and implemented the proposed SpaceWire router switch on the D-RMTP I SiP for parallel/distributed real-time control. We evaluated the basic performance of the proposed SpaceWire network. Now we are going to measure the real-time performance of the proposed real-time SpaceWire network, including the maximum latency and the average latency of each priority packet by RTL simulation. We will also measure them by using several D-RMTP I SiPs connected by the proposed SpaceWire network.

7. Acknowledgement

This research was supported by JST, CREST. This work was also supported in part by Grant in Aid for the Global Center of Excellence Program for "Center for Education and Research of Symbiotic, Safe and Secure System Design" from the Ministry of Education, Culture, Sport, and Technology in Japan.

8. References

1. ECSS, "ECSS-E-ST-50-12C, SpaceWire-links, nodes, routers and networks".

2. Christian Fraboul, Thomas Ferrandiz, Fabrice Frances, "A method of computation for worstcase delay analysis on SpaceWire networks", IEEE Symposium on Industrial Embedded Systems, Switzerland, July 8-10, 2009.

3. Nobuyuki Yamasaki, "Responsive Multithreaded Processor for Distributed Real-Time Systems", Journal of Robotics and Mechatronics, 2005.

4. Liu, C.and Layland, J, "Scheduling algorithms for multiprogramming in a hard real-time environment", Journal of the ACM, Vol.20, pp.46-61, 1973

5. Nobuyuki Yamasaki, "Responsive Link for Distributed Real-Time Processing" The 10th International Workshop on Innovative Architecture for Future Generation High-Performance Processors and Systems, pp. 20-29, January, 2007.

6. Shinpei Kato, Yuji Fujita and Nobuyuki Yamasaki, "Periodic and Aperiodic Communication Techniques for Responsive Link", 15th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, pp135-142, August 24-26, 2009.

7. Yuji Fujita, Shinpei Kato and Nobuyuki Yamasaki, "Real-Time Communication and Admission Control over Responsive Link", The IASTED International Conference on Parallel and Distributed Computing and Networks, pp. 131-138, Innsbruck, Austria, February 12-14, 2008.

SIMULATION OF A SPACEWIRE NETWORK

Session: SpaceWire Networks and Protocols

Long Paper

Authors : Philippe Fourtier, Alain Girard, Antoine Provost-Grellier, François Sauvage

Thales Alenia Space,

Etablissement de Cannes,

100 bd du Midi, 06156 Cannes La Bocca, France

E-mail: *alain-felix.girard@thalesaleniaspace.com, antoine.provost-grellier@thalesaleniaspace, Francois.Sauvage@thalesaleniaspace.com, fourtier.philippe@thalesaleniaspace.com* (*phone : + 33492926104)*

Abstract :
Data traffic gets more and more heavy and complex, mainly due to performance enhancement of payloads. The SPACEWIRE standard usage covers the growing need of high speed board-to-board communication links. This requires new tools providing efficient help in conception, development and validation of flight network designs. This paper presents a simulator dedicated to embed SPACEWIRE networks, taking into account all communication layers, up to constraints induced by nodes' behavior.

1. INTRODUCTION:

The article covers the development and set-in-use of a first validated version of a SPACEWIRE traffic simulator, intended to support Space missions.
Use cases have been selected, covering the various domains: exploration, LEO missions, robotics, interplanetary for main ones.

2. DEVELOPMENT OF A REPRESENTATIVE SIMULATION TOOL:

2.1 Development history:

When SpaceWire standard started to be the privileged solution for high rate communications, it appeared that classical validation means as bus monitoring, line analyzer or any other SW based tools were not adapted to SpW based networks, multiple point-to point links requiring too many measurement points. Furthermore, the traffic becomes more complex, taking advantage of much higher data rates. This led to specify and start the development of a representative tool making and executing numerical models of SpaceWire networks, where each exchange can be observed.

A study simulator called MOST (Modeling Of SpaceWire Traffic) has been first developed by TAS-F in year 2006, based on OPNET toolkit dedicated to network modeling.
The present version of the simulator proposes a library including the ESA SpW router, generic nodes, generic application models (Producer/Consumer).
Many features of the standard are implemented, including all basic SpW protocol (flow control, communication opening sequence, time-code, logical and physical addressing, local

buffering …) and upper layers like are GAR mechanism and RMAP protocol (Remote Memory Access Protocol) as described in ECSS-E-50-51.

MOST has been developed with respect to requirement to be an opened and living tool. An incremental development has been applied, each step enriching the simulator by modeling new SpaceWire features and appending SpW component.
In addition, MOST provides user with ability to insert functional behavioral models of nodes, without impacting standard SpaceWire models. A specific documentation is available for users in the form of manuals, each dedicated to the role of developers (product maintenance, models developers and network designers).

2.2 Concept:

2.2.1 Objectives of the simulator:

SpaceWire networks are much more complex than data bus or single node-to-node serial links, such that hand-made analysis or classical traffic analyzers are no longer adapted. Hence, the main goal is to have a representative validation tool dedicated to SpaceWire Network, aimed at supporting system and design activities in all phases of a project, from phase-A (i.e. pre-developments, trade-off) up to operational mission life (maintenance, ground investigations), by providing mean:

- To create quickly a numerical model of networks topology,
- To decrease design risks and secure planning, by early verification,
- To Keep control on traffic load, globally and in all parts of the network,
- To identify weak parts of the network topology,
- To verify load margins and transfer performances,
- To evaluate/refine budgets of resources, meeting system specifications,
- To verify the network FDIR by injecting simulated failures,
- To replay anomaly cases observed as test scenario,

2.2.2 Description:

Built upon the OPNET toolkit, a SpW library is provided, with validated up-to-date models of SpW parts (routers and nodes).
Components models are fully representative of the SpaceWire standard and of space qualified components of the market (i.e. ESA router, nodes), based on data sheets of suppliers.

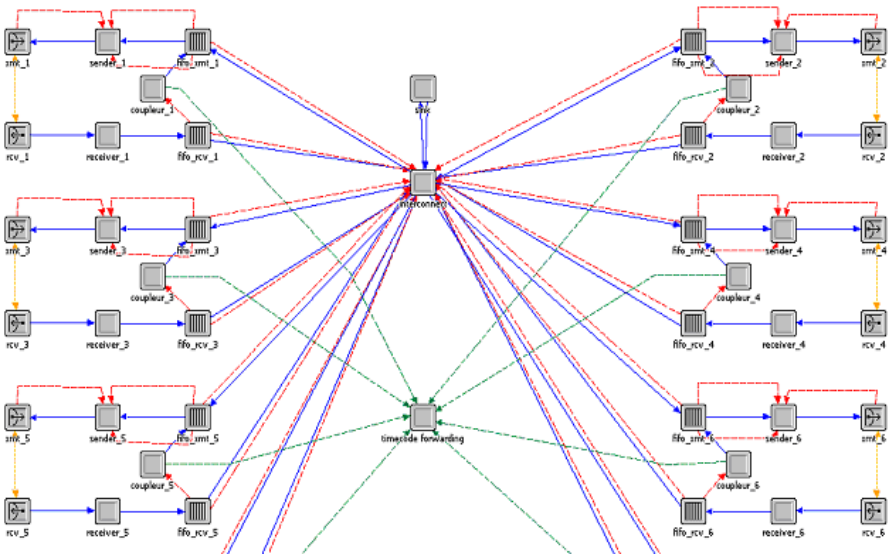


Figure 1: Standard model of router with time-code

The traffic simulation covers all layers of communication, starting with transfer of characters. It covers all standard features: flow control, time-out, time code, physical or logical addressing, GAR mechanism … up to application. Default applications are provided for nodes, based on simple data generation and consumption models.

MOST takes advantage of the OPNET Man Machine interface, to propose an easy and natural way to build custom networks. The nominal way to do it is to build components instances via a drag-and drop selection mechanism, then to link then together with wires as stated in mission design, and finally configure attributes of these links.

Once drawn and configured, scenarios are executed, recording standard or user-defined statistics, which can be drawn after execution in different ways. All or part of traffic can be observed, per wire and direction. Figure 3 represents the detailed traffic between 2 end users. The X axis drawing is time representative. Width of busy periods represents the duration of character transmission.

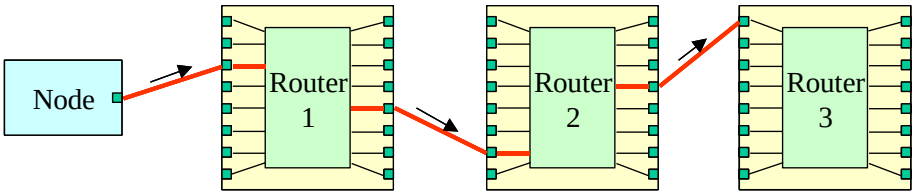


Figure 2: Network matching Traffic pattern of next figure

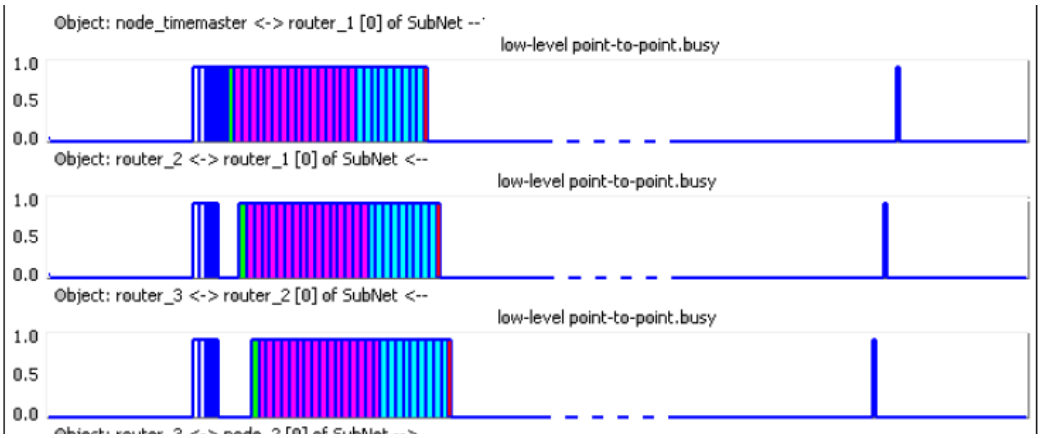


Figure 3: Graphical representation of traffic (node → router1→ router2 → router3**)**

The simulator is also required to cover anomaly cases, where numerical model is the unique way to check these unexpected events. To do so, pre-defined anomalies can be injected within scenarios simulating traffic loss, dumb node, erroneous characters, endless messages …

2.3 Two kinds of end users

2.3.1 Node Model Developers

These people are in charge of development of customized nodes. This work consists in replacing default applications by representative behavioral models of nodes. The standard part of models is not affected by these developments.
The developments are led under OPNET environment, using OPNET state machines and C language. A specific user manual is intended to model developers.

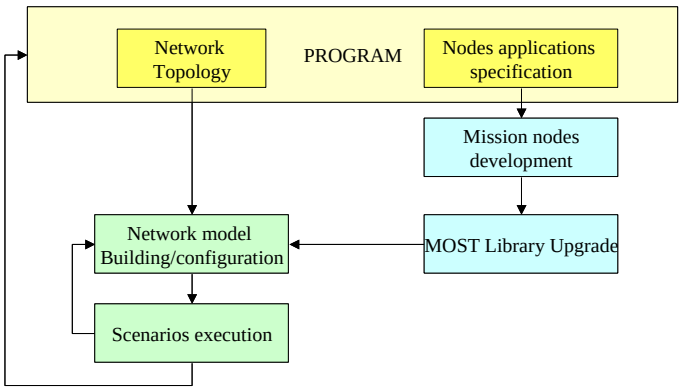


Figure 4 : MOST process

2.3.2 Network designers

They are in charge of implementing network designs by using the graphical editor, picking building blocks in library, linking them by drag-and-drop interface.

Then, they affect the attributes of blocks with mission values. At that time, they are ready to run scenarios. A specific user manual is intended to network designers.

2.4 An open tool:

2.4.1 Evolution (adding / enhancing components):

Upgrade of the simulator can be done at several levels:

- Evolutions of the standard can be easily inserted, due to the layered and structured modeling approach (sender, transmitter, router function, time-code generator and others are clearly identified and separated bricks within models. Furthermore, transport layer is clearly separated from the protocol and upper application layers.
- New components can be added, using a building blocks approach (bricks re-use).
- New protocols can be easily added, by switching bricks on the dedicated layer.

This task is under responsibility of the development team in charge of maintaining the generic SpaceWire simulation tool.

2.4.2 Customization to mission:

Mission designers can build their network topology, configuring all links, but also:

- Nodes models let users customize application behaviors, introducing data generation and consumption models.
- Custom models can be enhanced anytime, such to follow evolutions of the embedded design (incremental approach, refining models).

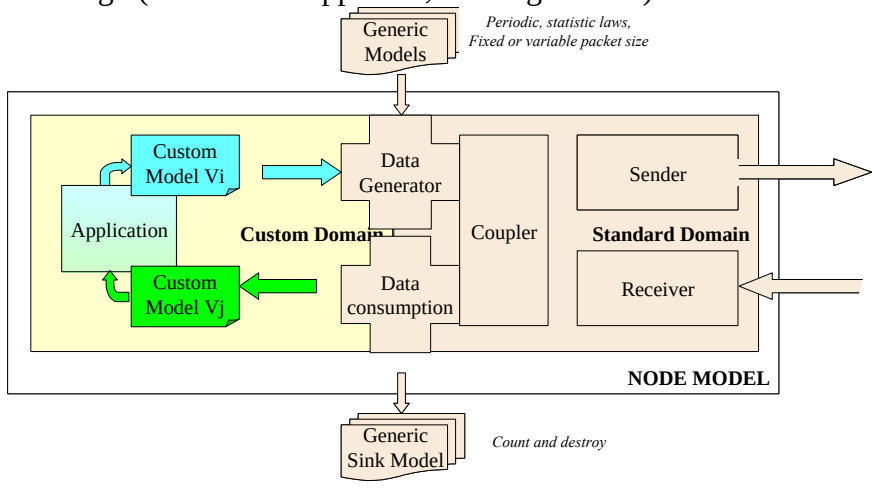


Figure 5: Node is composed of a standard part and a customized application

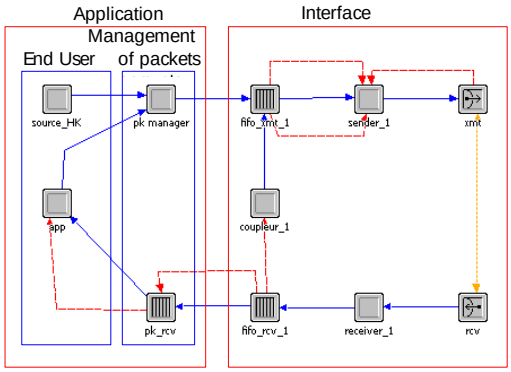


Figure 6: Architecture of node matching concept of Figure 5

Node models can be created such to represent a board function: computer, mass memory, instrument, or any other platform or payload unit. Models can be re-used in other missions (library is enriched each time). This task is realized in OPNET layered environment (see Figure 7), under responsibility of *Node Model developers*. Custom statistics can be defined, attached to these models to be selected at simulation time.

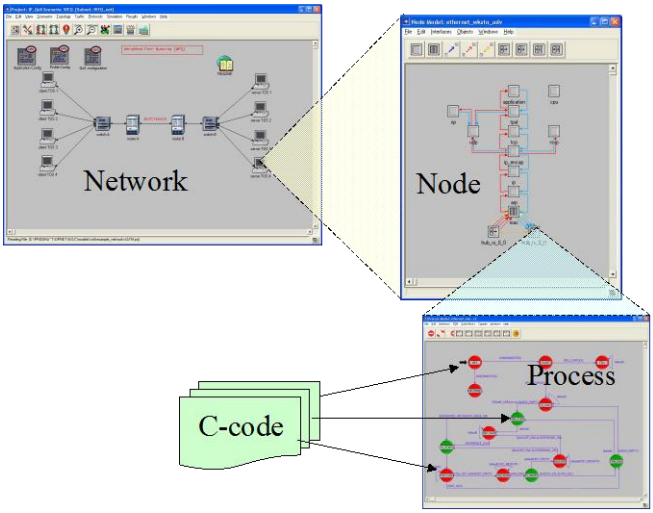
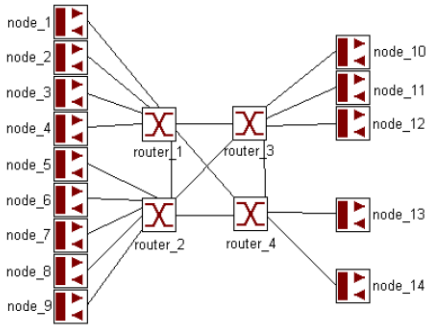


Figure 7: OPNET design layers

2.5 Simulator usage:



The objective of the simulator is to experiment and verify network designs, by executing mission representative scenarios of traffic load. When the network topology is specified, the designer builds the network model, picking the components in the SPACEWIRE library (including custom nodes) and connecting them by using the graphical editor.

Each network item (node, router and links) can be parameterized, setting values in a pre-defined set of attributes (data rates, routing table, buffer sizes etc …).

Figure 8: SpW network topology (example)

The network model can then be experimented within various scenarios, simulating activities of nodes as sources of traffic. This task is under responsibility of the *Network designer*.

2.6 Expected Results:

At execution time operators select which statistics will be logged.
Traffic analysis and resources consumptions are the main issues of simulation sessions, such to show margins, traffic jam or data loss, effects of anomalies, memory consumption or any other custom statistic attached to specific nodes.
The drawings, created from recorded results, either represents macroscopic statistics or accurate traffic.
Reporting helps designer in verifying the protocol efficiency, the design performance and margins, in refining budgets and identifying slowing sources.

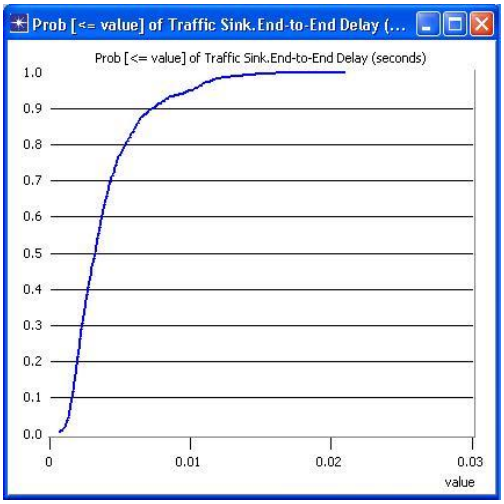


Figure 9: statistics drawing example

An advantage of the simulator is its ability to restitute the traffic of any part of the network, even in contingency cases, which is impossible on actual HW.

Main Benefits are:
<u>At design level:</u>

- To Keep the complexity of SpaceWire traffic under control, by an accurate modeling,
- To get a quick return on design weakness or risks, identifying critical items,
- To verify margins and performances,
- To evaluate resources budgets (buffer sizes). This can be done by 2 ways:
  - infinite resources, showing the exact maximum need
  - sized resources, showing margin or raising error in case of overflow.

<u>At programmatic level:</u>

- To provide a continuous help in all phases of the development (from phase A to phase E), thru nodes models refinement.

<u>In validation:</u>

- To execute contingency cases or to replay tests found faulty on actual HW.

3. SIMULATOR'S VALIDATION

A particular effort has been put on an incremental test approach, starting with simple use cases of easy understanding, mainly verifying compliance to standard. Cross validation has been done by comparing HW provided results to simulator's.
Some use cases have been selected for functional validation and domain coverage, according to their representative features, concerning functions, architecture and network complexity.
A selection of these tests can be used as regression tests at each release of the simulator, in particular when adapting the simulator to new versions of OPNET environment.

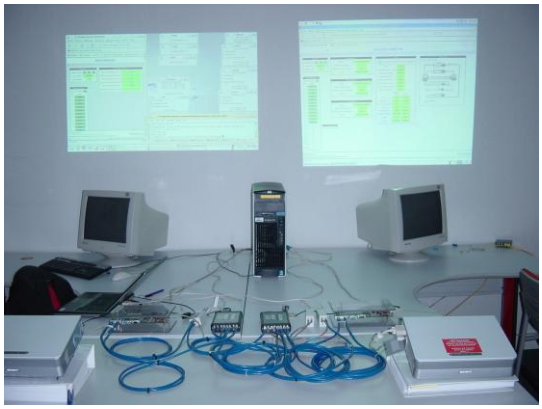


Figure 10 : SW/HW Cross validation

4. APPLICATION DOMAIN

4.1 Expected coverage

The objective is to dispose of a tool able to cover all space applications.

- all families of orbits or trajectories, LEO, MEO, GEO, Lagrange point, inter-planetary, landers, rovers and robotics.
- All domains: observation, science, exploration, telecommunication, etc…

That's why next work covers simulation of 4 typical and realistic use cases, respectively belonging to interplanetary, LEO observation, exploration and robotics domains.

4.2 First use case: interplanetary missions

This use case was proposed for an orbiter, saving science data from multiple instruments within a mass memory, and restituting these later on in X-band or Ka-band TM channels. The interesting point is that some auxiliary data have to be inserted within the science flow, which is done by the SSMM controller (Mass Memory) or PM (Processor Module). This kind of features is impacts fluidity of the traffic on the network.

Figure 11 is a limited scenario with science flow (in red) and C/C flow from PM (in blue).

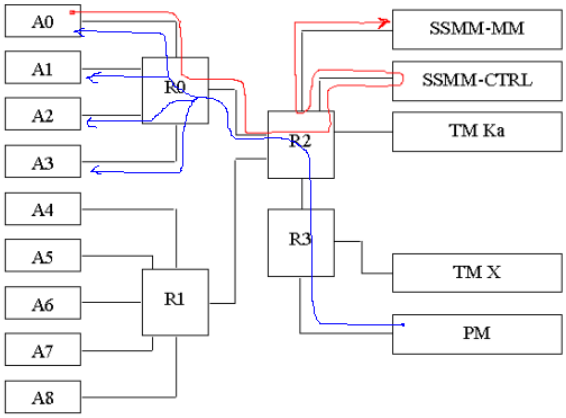
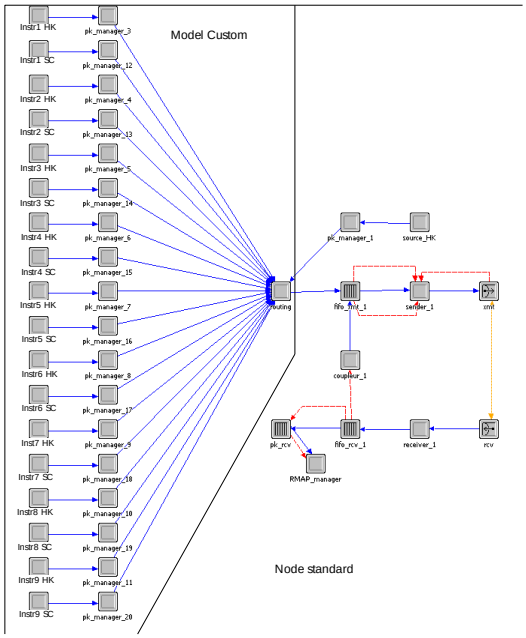


Figure 11 : Network for orbiter with 8 instruments A1 to A8



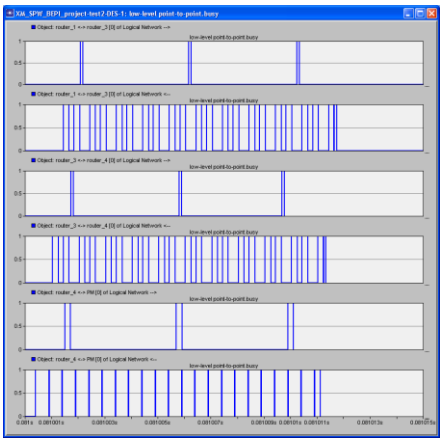


Figure 12 : Small extract of results obtained for scenario of Figure 11

Figure 13 : Custom model of the SSMM controller node

4.3 Second use case: LEO observation

This use case is Sentinel-3 application, involving 4 instruments, a PDHU as mass memory and the central computer of the platform. This is a classical configuration of embedded network.

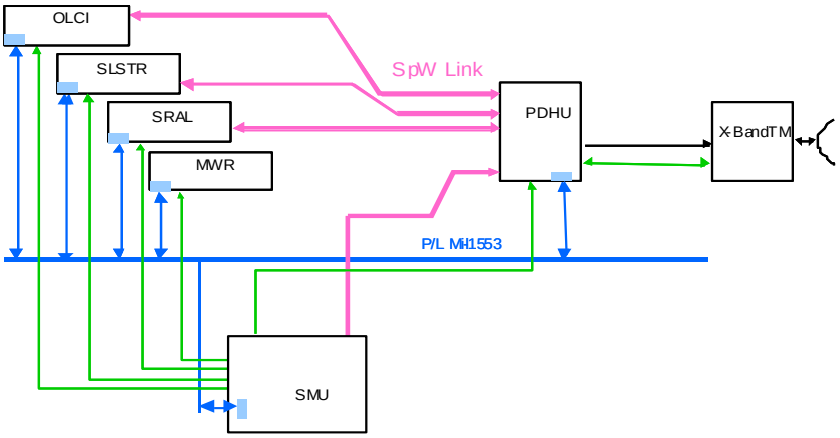


Figure 14: Sentinel 3 SpaceWire network

4.4 Third use case: exploration domain

The network of Figure 15 is issued from soft autonomous landing mission, requiring sensor data in a continuous way, at a high rate from specific complex sensors. The data load is far beyond the performance of classical data bus, as it requires imaging units (optical or LIDAR). The fail-op nature of the landing phase requires hot redundancy, making double complexity for the network.

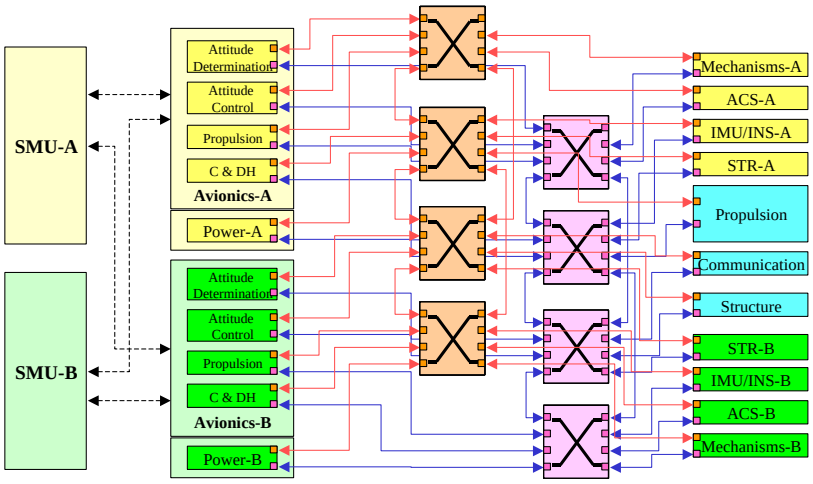


Figure 15: Typical exploration use case - Soft landing mission

4.5 Fourth use case: robotics

The network on Figure 16 is issued from robotics space application, requiring visual monitoring sensors and motors encoders as actuators.

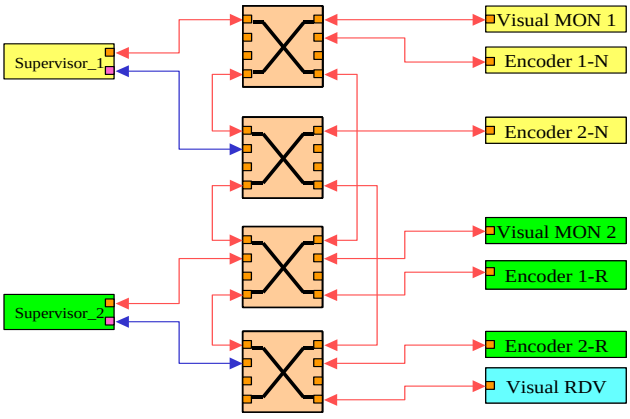


Figure 16 : Space embedded Robotics network

5. AND NEXT:

Taking advantage of new modules of last releases of OPNET environment, TAS is now on the point to investigate the possibility to connect the numerical simulator to HW sub-network, making a numerical / HW hybrid test configuration.
The main objectives are :

- to start early with verification of functional behavior of actual HW, even partly,
- to save planning in case of late delivery of design parts,
- to replace some part of the network by models, injecting easily traffic anomalies to verify contingency cases, or to check the behavior of actual HW in these cases,

Furthermore, TAS has been selected by ESA for a study, aimed at provide a SpaceWire simulator, which requirements cover widely the MOST TAS perimeter. SOW requires explicitly to study feasibility of hybrid simulation.

6. CONCLUSION:

TAS expressed the need for a SpW simulator in early 2006, with questions raised on validation of embedded network, considering such design more complex than classical data bus or single node to node links.
This led to the development of the first version of in-house simulator MOST, enriched later on with time-code and more complex features as GAR or RMAP. OPNET environment was chosen, because dedicated to network simulation, and because of its Man Machine interface.

The simulator has been experienced on several use cases, in the frame of nominal and contingency scenarios. Results have shown the usefulness of such a tool, and MOST could be the root for industrial simulation product.
This supposes to enhance models, to add new building blocks, to complete the formal validation with reference use cases, to complete documentation and finally ensure the maintenance, by in particular following main release of OPNET environment.

A door is opened towards the hybrid network test bench, involving HW sub-networks and numerical sub-networks. Such approach looks possible with modules of more recent OPNET versions, but requires additional feasibility analysis.

The growing implication of SpaceWire standard in board-to-board communications since first version of MOST has never been denied, even if slower than expected. The enhancement of performances of embedded avionics (more resources) lead to more ambitious performance requirements, including higher data rates, hence more complex communication networks and greater volumes of exchanged data.

Validation means shall follow this evolution to keep control and quality of designs of more complex embedded systems, communications being a critical issue. Numerical simulators are a part of the response, as HW means will be expensive and difficult to set in use early.

STAR-LAUNCH AND NETWORK DISCOVERY

Session: SpaceWire Networks and Protocols

Long Paper

Stuart Mills, Chris McClements

STAR-Dundee, c/o School of Computing, University of Dundee, Dundee, Scotland, UK

Steve Parkes

University of Dundee, School of Computing, Dundee, Scotland, UK

E-mail: stuart@star-dundee.com, chris@star-dundee.com, sparkes@computing.dundee.ac.uk

ABSTRACT

STAR-Launch is a new software tool that can be used to launch applications and modules to interact with STAR-Dundee SpaceWire devices. It will provide the ability to discover devices on a SpaceWire network and display these graphically. The software will then allow applications and modules to be launched to perform operations specific to the devices discovered.

This paper describes the features that are currently provided by the STAR-Launch software, and the new features which are to be added. The updated software will provide many of the features described in the draft SpaceWire-PnP Protocol Definition. The services in the draft PnP definition that will be implemented in STAR-Launch are described in this paper, along with details of our experience of implementing these services.

1 INTRODUCTION

STAR-Dundee has been working on an updated API for accessing SpaceWire devices. The new API provides a common interface for accessing all STAR-Dundee SpaceWire devices [1], regardless of whether they are routers or interfaces, and whether they are connected by USB, PCI, or some other mechanism. It also provides a number of components for implementing common functionality, such as RMAP [2] initiators and targets. Components are also included to create virtual devices and links, which allow virtual representations of SpaceWire networks to be built in software [3].

To allow users to access the new features provided by the updated API, new applications such as STAR-Launch have been developed. The main purpose of STAR-Launch is to display the SpaceWire devices connected to a PC, and allow applications to be launched to access these devices. It can display both physical devices as well as virtual devices, which can be created using the application.

STAR-Launch is being extended to allow the SpaceWire network to which the local physical devices are connected to be discovered. This will make use of the Plug and Play (PnP) functionality described in the latest draft of the SpaceWire-PnP Protocol Definition [4]. STAR-Dundee devices will be modified to support the SpaceWire-PnP Protocol, while software modules will be added to the STAR-Dundee Application Programming Interface (API) to provide PnP support in both applications and virtual devices.

This paper describes the features of STAR-Launch and the services from the SpaceWire-PnP Protocol Definition which will be implemented. Comments on the Protocol Definition and details of the experience of implementing the services are also provided.

2 STAR-LAUNCH FEATURES

The current release of STAR-Launch automatically detects all STAR-Dundee SpaceWire devices connected to the PC on which it is running and displays an icon for each device. The user can run the default software application for a device by double-clicking on its graphical representation, or bring up a menu showing all available software for the device by right-clicking on it. An example screenshot is shown in Figure 1.

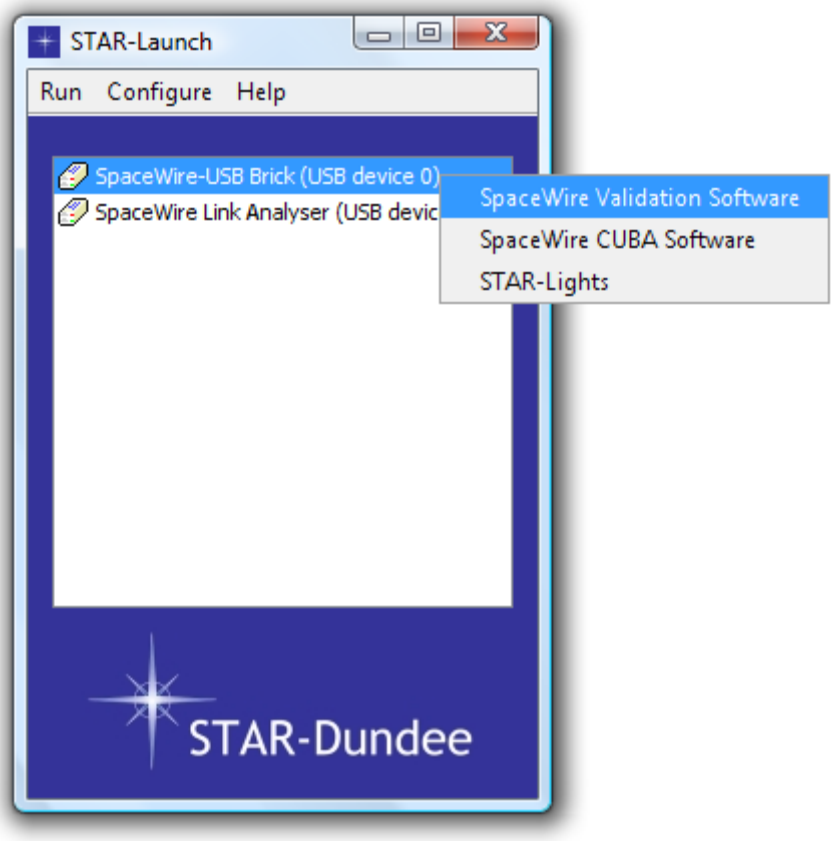


Figure 1: STAR-Launch Screenshot

The applications provided for each device type, and the default application for a device type, are all configurable by the user. For example, double-clicking on the

icon for a SpaceWire Link Analyser would normally run the Link Analyser software for that device. Double-clicking on a SpaceWire-USB Brick icon could bring up a window in which the Brick can be configured. Right-clicking on the icon will bring up a list of software which can be run for the SpaceWire-USB Brick, which could include the Validation Software, CUBA Software and the default option to configure the device. If a user has written some software which uses the SpaceWire-USB Brick, they can add this to the Brick's menu, and possibly set it as the default option.

2.1 VIRTUAL DEVICES

Following the addition of virtual devices and links to the STAR-Dundee API, STAR-Launch was updated to display all virtual devices and to allow new virtual devices to be added and existing virtual devices to be deleted. Some method to establish virtual links between virtual and physical devices, and remove these links when no longer required was also required. STAR-Launch is the ideal application for these purposes, so was extended to provide these features. Rather than just displaying a list of devices, STAR-Launch now displays the topology of a PC's virtual SpaceWire network, showing the links between applications, virtual devices and physical devices.

Virtual devices are treated in a similar manner to physical devices in both the STAR-Dundee API and STAR-Launch. STAR-Launch allows applications to be run for virtual devices and virtual devices to be configured, just as it does for physical devices. STAR-Launch also allows Virtual Link Analysers to be attached to virtual links, so that traffic crossing over a virtual link can be monitored and recorded. This is a powerful debugging tool and can be used not only in virtual SpaceWire networks, but also to view the traffic passing between a physical device and an application, for example.

2.2 NETWORK TOPOLOGY

As STAR-Launch provides the ability to interact with both physical and virtual networks, and also displays the local virtual network, it was decided to expand this functionality to display the physical SpaceWire network or networks connected to a PC. This would allow the user to see the topology of the physical SpaceWire network, configure remote devices, send packets or commands to these devices, etc.

In order to display an arbitrary physical and/or virtual network, a method to discover the network is required. The University of Dundee and STAR-Dundee have researched methods to discover networks, and the draft SpaceWire-PnP Protocol Definition includes the results of some of this research. As this protocol definition aims to standardise network discovery, device configuration, etc. the PnP Protocol is the obvious choice to provide this functionality in STAR-Launch.

3 SPACEWIRE-PNP SERVICES

The STAR-Launch application is being updated to include support for the SpaceWire-PnP Protocol. The first stage of this process is to add support for the protocol to the STAR-Dundee API. The API already provides RMAP target and initiator components, which can be used to implement RMAP support in software. This functionality is being extended to provide PnP initiator and target components in the API. This will allow applications to discover networks and configure devices, and

will allow virtual devices and applications to be implemented which can be discovered and configured using PnP.

The STAR-Dundee API will provide all the services outlined for both levels defined by the SpaceWire-PnP Protocol Definition, Level 1 and Level 2, and will allow both active and passive nodes to be developed. Active nodes are those nodes which may act as the initiators of PnP commands, such as the nodes which request the properties of other devices on the network. Passive nodes are those that only act as a target for PnP commands.

The services for both Level 1 and Level 2 that are being implemented in the API are similar, the difference being that Level 2 services must be able to cope with multiple active nodes. An example of when Level 2 services are required is when there are multiple nodes trying to discover the network at the same time. The services to be provided are described briefly below.

Device Identification: This service allows a device to identify itself and to describe its characteristics.

Network Management: This service provides methods to discover and manage a network.

Link Configuration: This service can be used to query and configure the properties of the links on a device.

Router Configuration: This service is provided by routing devices to allow properties specific to a router to be queried and configured.

Time-Code Source: This service is optional and can be used to enable and configure a time-code source.

There are currently two SpaceWire-PnP Capability Services defined by the PnP Protocol Definition. These Capability Services are protocols that a node supports for transporting data. The STAR-Dundee API will provide support for both of these services, which are described below. As with the other PnP services, these are being built on top of the RMAP target and initiator components already provided by the API.

RMAP Data Sources: A target which supports this service can produce data in response to RMAP read commands, while an initiator can produce data using RMAP write commands.

RMAP Data Sinks: A target which supports this service can consume data in RMAP write commands, while an initiator can consume data by initiating RMAP read commands.

4 Network Discovery and Device Configuration

Once the PnP services have been added to the STAR-Dundee API, they will be incorporated into example virtual devices provided with the API. It will then be possible to discover these virtual devices if they are connected in a virtual network. If the virtual network is also connected to a physical SpaceWire network, it will be

possible to detect the virtual network from any point on the physical SpaceWire network. The virtual network will appear as just another section of the SpaceWire network, and the active node discovering the network will treat the virtual devices in the same way as the physical devices.

The updated STAR-Launch application with support for the PnP services will allow the SpaceWire network to be discovered and configured. It will be possible to discover and configure existing virtual devices on other PCs connected to a SpaceWire network. This will allow individual PCs to be used to represent different subsystems, for example, and each can be configured from a single location.

In order to enable discovery and configuration of physical SpaceWire devices using the SpaceWire-PnP Protocol, these devices must support the PnP Protocol. STAR-Dundee's routing devices, the SpaceWire Router-USB and SpaceWire-USB Brick will be updated to include support for PnP. It is already possible to configure links routing tables, etc., of these devices over a SpaceWire network using RMAP, but they will be updated to use the register mappings defined for SpaceWire-PnP and to use the SpaceWire-PnP Protocol ID [5].

The SpaceWire-PnP software components added to the API can be used to provide a PnP service for devices. To support SpaceWire-PnP in SpaceWire interface devices, such as the STAR-Dundee SpaceWire PCI device, no modification is required to the hardware, as shown in Figure 2. This diagram illustrates how configuration port traffic received by the PCI device (packets with a first byte of 0) could be routed to a protocol dispatcher by a virtual router. PnP traffic could then be routed to the PnP service by the protocol dispatcher, while RMAP traffic could be routed to an RMAP implementation of a configuration port. Meanwhile, packets with a logical address of 70, for example, could be routed from the PCI device to a mass memory unit application.

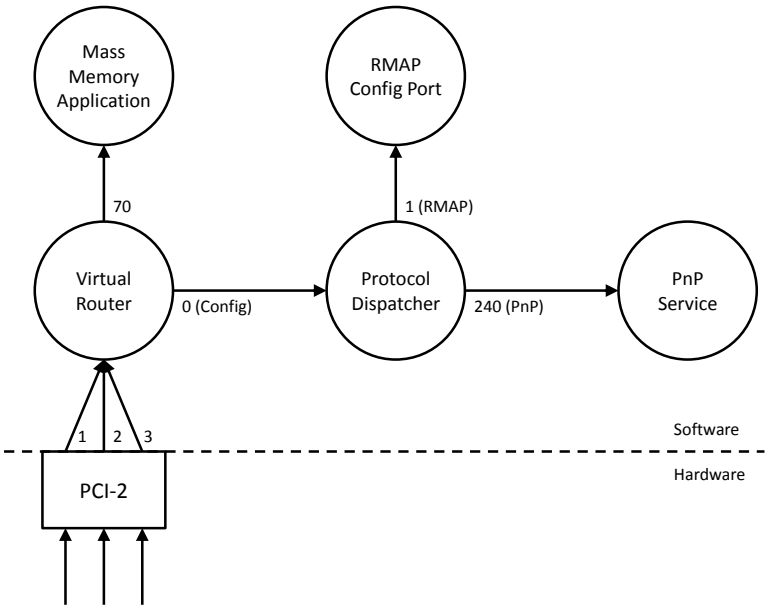


Figure 2: SpaceWire PCI Interface Device Supporting PnP

This diagram could be simplified by removing the protocol dispatcher and RMAP configuration port. The PnP service would then deal with all packets sent to the configuration port, ignoring any with an incorrect protocol ID.

Alternatively, if there is a virtual SpaceWire network connected to an interface device, it may be required that the device does not respond to PnP requests. It then becomes a pass-through device which is not visible to the active nodes which detect the network. The instance of STAR-Launch running on the local PC will have knowledge of the virtual network, and will display it in its network topology display, as shown in Figure 3, which shows three applications connected to each of the links of a PCI device. STAR-Launch instances running on other PCs will have no knowledge of the PCI device's existence, and will just see three nodes connected to each of the links, if these applications respond to PnP requests.

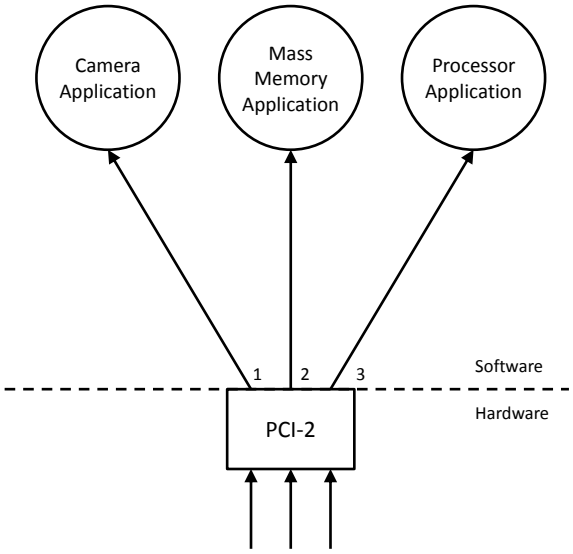


Figure 3: SpaceWire PCI Interface Device Acting As a Pass-Through Device

5 DEVELOPMENT EXPERIENCE

The implementation of the SpaceWire-PnP features in the API, the improvements to STAR-Launch to support these features, and the addition of PnP support to STAR-Dundee devices, are ongoing. The experiences of the developers working on these features can be of use to others working on SpaceWire-PnP developments.

The developers found that it is quite simple to implement the protocol using existing RMAP components, as SpaceWire-PnP is built on top of RMAP. STAR-Dundee hardware was already designed to allow properties to be read and configured using RMAP reads and writes, so software and hardware RMAP implementations were already present.

The SpaceWire-PnP Protocol Definition is very long (211 pages), and this gave the impression that the PnP Protocol was very complicated. However, after some initial reading it became clear that there are only around 20 pages that everyone working with SpaceWire-PnP must read: sections 3 and 4. The sections which follow this,

sections 5, 6, 7 and 8, provide all the technical information to implement a service, and are useful reference points for the developers.

The protocol definition is not yet at a stage where it can be considered complete. In addition to the prototyping and technical verification required before SpaceWire-PnP can be considered a suitable solution, there are still a number of typographical improvements that need to be made to the document. It does however appear to provide a capable solution to provide Plug and Play support over SpaceWire.

6 SUMMARY

This paper has described the features of STAR-Launch, the features of SpaceWire-PnP, and how these two are being combined. The experience of the developers working on the implementation has shown that the SpaceWire-PnP Protocol Definition, while not yet perfect, seems to provide a comprehensive set of services to allow networks to be discovered and devices to be configured.

7 REFERENCES

1. STAR-Dundee, http://star-dundee.com/products.php, STAR-Dundee SpaceWire Products, STAR-Dundee Website.

2. ECSS, "SpaceWire – Remote Memory Access Protocol", Standard ECSS-E-ST-50-52C, Issue 1, European Cooperation for Space Standardization, February 2010.

3. S. Mills, A. Mason, S.M. Parkes, "STAR-Dundee Virtual Devices and System Simulation", International SpaceWire Conference 2010, St Petersburg, Russia, 22–24 June 2010.

4. P. Mendham, A. Ferrer Florit, S.M. Parkes, "SpaceWire-PnP Protocol Definition", Issue 2.1, University of Dundee, September 2009.

5. ECSS, "SpaceWire Protocol Identification", Standard ECSS-E-ST-50-51C, Issue 1, European Cooperation for Space Standardization, February 2010.

FRAMING IN SPACEWIRE NETWORKS

Session: SpaceWire Networks and Protocols

Short Paper

Suvorova E. A., Sheynin Y.E., Pyatlina E.A.

St. Petersburg State University of Aerospace Instrumentation
67, Bolshaya Morskaya st. 190 000, St. Petersburg RUSSIA

E-mail: E-mail: sheynin@aanet.ru, suvorova@aanet.ru

ABSTRACT

SpaceWire protocols use packets at the Network level and flits for flow control at the Exchange level. For further evolution of SpaceWire technology framing could be introduced. The article considers different forms, levels and application of frames, considers adaptability of different variants of frames implementation for SpaceWire. As a trend frames are correlated with virtual channels. Reasons for extending frames from the traditional Data Link layer to the Network layer are considered in correlation with basic SpaceWire features that have shown their efficiency.

1 INTRODUCTION

Classical definition attributes frames to the Data Link layer, [i], as the PDU (Protocol Data Unit) of this layer. According to the OSI Reference Models layers *messages* are segmented at the Transport layer in sequences of *packets*, which are transmitted and routed by the Network layer; packets are reformed into *frames* at the Data Link layer. Relation between a packet and a frame could be different – the packet could be enveloped into a frame, or it can be segmented further and transmitted over the data link as a sequence of frames. Modern interconnections use further levels of segmentation and units also: *flit* (flow control units) and *phit* (physical unit – unit of information that is transmitted by the data link in clock period), [ii].

Typical frame has a header, payload field and check sum (usually CRC). Frame format and length of a frame can depend on the frame type; e.g. data frames and control frames. for a particular type it is usually fixed or limited by a fixed value. The frame header holds information on its type, priority, length and some information that attributes the frame to a particular packet. Some standards include additional information to be used by the Data Link layer or by higher protocol layers; e.g. in FibreChannel the frame header tells if the frame is the first or the last in the frames sequence for the message; for connection-oriented services it also marks if the connection should be closed after this frame. The number of frame in the frame sequence could be included in the header to support in order delivery.

The SpaceWire has a typical flit-based flow control mechanism at the data link layer (though the word flit itself isn't used in the standard). Packets are specified as basic

PDU. Messages, frames are not used in the SpaceWire standard. A message as the Transport layer SDU (service data unit) is either to be enveloped in a single packet, or to be segmented into packets somewhere out of the specified Transport layer. Such vision is supported by unlimited length of SpaceWire packets; no packet length limit is given in the specification, all the layers' protocols are specified to be operational at any length of a packet (though a packet should have final length and be finished by EOP/EEP). The standardized SpaceWire wormhole routing helps to disjoint a routing switch buffering and a packet length. However, it fixes dependency of a routing switch input/output ports occupation, as well as internal switch data paths, on the routed packet length. Once a packet started to run by the link, by the switched in the router input-output ports pair any other packet transfer that needs the same input or output port would have to wait finish of the current packet transfer to make its attempt to access the port. Longer the current packet is, for longer time it will occupy the link. It makes hard to ensure faire distribution of links and routers throughput between different packet flows, to support prioritized packet transfer, give reliable estimation of packet delivery time, build QoS transmissions. Clever exceptions in SpaceWire are control codes, e.g. time-codes, that preemptly cut in a data bytes flow.

Some standards, workaround the problem by strictly limiting a packet length (e.g. 276 bytes in RapidIO); thus any packet wouldn't occupy any path for a long period. Another way is to apply to basics of the OSI Data Link layer and introduce extra level of segmentation – segmentation of packets into frames. Thus unlimited in its length SpaceWire packet would be cut into quite limited in length portions, which would be formatted into frames, and sent over the link as a sequence of frames. If we could attribute a frame to a particular packet in some way, then we can multiplex frames from different packets, make several packets to run simultaneously over one data link, intermitting by frames. Such a way is proposed, for instance, in the initial SpaceFibre specification as an evolution of SpaceWire, [iii]. Introducing frames, correlating it with SpaceWire features we get new facilities to solve several types of problems.

2 FRAMES

2.1 FRAMES AND RELIABLE DELIVERY IN A DATA LINK

Unlimited length of SpaceWire packets gives no way for reliable delivery at the Data Link layer; it would. require buffering of the whole packet to be able to retransmit it in case of delivery with errors or loss. A limited length frame could be buffered, could be checked on delivery and retransmitted in case of loss or an error on receipt. Typically, buffering is required in the output link, at the transmitter side and could be used or not at the receiver side (a receiver could check the incoming frame on the fly).

2.2 FRAMES, FLOW CONTROL AND CREDITING

Flow control is one of the key features of the Data Link layer that is implemented in most network standards. Different modes of flow control could be used: crediting by receiver or by transmitter, absolute or relative credits, without crediting – deleting a received packet if there is no space for it at the receiver side and indicating later retransmission request (e.g. in RapidIO). In a data link the SpaceWire uses flow control with crediting by the receiver, flit (8 bytes) based. The flit isn't correlated with packets length and packets' boundaries. Thus a packet end (e.g. 4 bytes) could jam in

the link for a while due waiting for a full flit credit. Should we move to a frame-based crediting (proposed by the SpaceFibre draft) or leave the flit-based crediting? It is correlated with features in protocols and implementations, e.g. with buffering.

2.3 FRAMES AND BUFFERING

The SpaceWire protocol stack architecture is based on the assumption that a packet need not be buffered in a router it passes. The wormhole routing, a type of cut-through routing, is standardized. Thus a router buffer space is not related with a packet length. Rather it is correlated with data link layer PDU size and flow control units, 8 byte flits. Definitely we should like to save this feature in further evolution. The classic SpaceWire data link PDU is a packet; the flit is 8 byte chunk of information flow. With introduction of frames the frame will be the data link PDU. Should we move all the buffering to this type of PDU? It is a question for investigation. As one can see from the above paragraphs we need PDU-frame buffering in the transmitter part of the data link if we want to ensure reliable delivery at the data link layer. However, the flow control could be either moved from flits to frames (the SpaceFibre draft way), or left at the flit layer of the classic SpaceWire. In the latter case input buffer size could be smaller. We should also take into account that buffer size could depend upon implementation of another features of data link protocols, e.g. o possible sliding window flow control and in-order delivery of frames of the packet. Estimating known in networks theory and practice methods one should keep in mind the basic principal of the SpaceWire technology – compact, low cost implementation.

2.4 FRAMES AND PRIORITIZED PACKETS DELIVERY

By multiplexing by frames multiple packet simultaneous transfer over a single data link we open possibilities for fine grade prioritization in packets delivery also. In SpaceWire a prioritization could be implemented only at the packet level. With unlimited packet length it looks rather poor: any low priority packet, which happened to get a link while there were no high priority ones, can hold the link for a long time. Add here also no obligation of a sender to form and send a packet in high rate and without gaps. With frames we can improve the situation by passing priority to the packet's frames and using it for link access arbitration in frames multiplexing. Frames from high priority packets would wait for a limited time for the link access (frame size is limited, unlike the packet size). It will make prioritized packet transfer reasonable. It also opens a way for predictable delivery time for high priority packets, though such time prediction calculations in many cases would be not an easy task.

2.5 FRAMES AND QOS

Further improvements in services provided by SpaceWire interconnections could be achieved by using frames for QoS. A packet source could attribute a QoS class to the packet; the class is passed to its frames. Any passed by the packet's frames flow data link and routers could analyze this class in all the parsing frames and use it in selection of priority and routs for frames forwarding. Frames features, its limited length and explicit indication of QoS it requires gives embedded in the interconnection algorithms to support QoS in a guaranteed way, not dependent of faire operation of packet sources and sinks throughout the interconnection.

2.6 FRAMES AND VIRTUAL CHANNELS

A general issue in framing is attributing frames to packets or messages. For attributing a frame some information in its header is used. It can be its parent packet or message identification, or attribute of a logical path. Frames are a typical mean in virtual channels implementation. Virtual channels share a physical link by multiplexing their flows in it. With frames it would be a fame-level multiplexing. It ensures independent operation of source-sink pairs that correspond to separate virtual channels that is helpful in solving many types of problems in robust network operation.

2.7 FRAMES, VIRTUAL CHANNELS AND CLINCH PREVENTION

Virtual channels can help in clinch prevention for duplex data link layer. A dense PDU output flow in one link direction could occupy the entire output buffer and leave no space for control PDUs that should be sent for incoming PDU flow protocol state transition at the other side of the data link. In case of similar situation at the opposite side of the link one would have a classic clinch. By assigning a separate virtual channel for control PDU transfer one guarantees that its link access would be granted regardless data PDU flow intensity. For instance, the Infiniband assigns a separate virtual line #15 for control PDU transfer, [iv]. In SpaceWire the Control codes help to bypass such danger by immersing the control to the lower layer PDU – control codes (Symbol level PDU), that transparently cut through a data characters flow. However, higher level protocols could require control packets exchange to support protocol's FSM operation; frame based virtual channels could be used to guarantee it.

2.8 FRAMING AND THE NETWORK LAYER

Frames multiplexing could be useful not only in a data link, but at the level of a routing switch also. SpaceWire supports multi-rate links operation and a router can have links that operate with different rates. If a slow link receives a packet that should be passed to a fast output link, the low rate input link would drive the rate of the high throughput output link; filled by NULLs gaps will be at the link, while no other ready packet could use the link until the current packet will run out. Input buffering packets could help, but it requires setting constraints on the packet length (to have reasonable buffers) and increase packet transfer latency. Frames could be buffered at router input ports; streams of frames from different packets could be multiplexed in the output port and run multiplexed at the output link, without gaps. However, it means that several packets (their parts – frames) simultaneously run from a link to the next router (or node) input port. Demultiplexing of these frames from different packets in a stream requires attributing frames to packets; attributed frames of different packets should be routed to appropriate buffers or the routing switch paths to appropriate output ports.

Attributing a frame to the packet could be based on some packet identification in the frame header. By joining frames with virtual channels the identification could be done by the virtual channel number thus assuming that only one (framed) packet is transmitted at a moment in the virtual channel. Virtual channels number is typically small (16 for Infiniband, 256 in the SpaceFibre proposal). It is good for compact virtual channel identification in frames and its processing in a router, but limits the number of packets that could run simultaneously multiplexed by frames. With a virtual channel identification, VCid, in the frame header a frame could be easily attributed to the virtual channel and thus – to the packet. For the SpaceWire Network

level, the first frame of the packet should be unpacked and the packet header used for routing; after determining the output port and switching path to it the frame is forwarded to the output port for transmission; next packet frames could be send by this path directly from the input port.

Above it is assumed that there is a set at the input data link and a set at the output data link. Appropriate virtual channel number is coded in the header of every incoming the router frame. The output port for the frame is determined by the packet routing, to which the frame belongs to. However a virtual channel for the frame at the output port should be determined also; the in-router switch path should connect not just ports, but the input port virtual channel with the output port virtual channel. Building this pair becomes a part of the routing switch procedure on receipt of a packet header (inside its first frame). Thus frames and virtual channels could be introduced in SpaceWire along with its efficient and low latency wormhole routing. However we should take into account necessity of a frame transformation in its path from the input to the output port: its input port VCid should be changed for another VCid of the virtual channel output port. Changing a byte in the frame header can lead to CRC re-calculation. Some standards take the part of the header, which is changed at a hop, out of the CRC coverage (e.g. Infiniband, [iv] and RapidIO, [v]). Additional transformations should be done in the first frame of a packet in case of a path or a regional address: header byte deletion changes the length and CRC of the frame.

The claimed above relation: frames of one packet only running in the virtual channel, could be also argued. With this assumption frames could be attributed to packets simply by the virtual channel number. Otherwise, if a virtual channel could have a frames mixture of different packets simultaneously, additional coding for attributing a frame to the packet is required in the frame header. This type coding in a frame header and its processing in every passed by the frame router is used in some standards (e.g. Infiniband, FibreChannel, [vi]). However, implementation cost for such complication should be estimated carefully in reasoning on using it in SpaceWire evolution.

3 CONCLUSION

Frames and virtual channels are efficient way for SpaceWire technology evolution, help to solve its problems and reduce bottlenecks. However to use the framing and virtual channels potential one could not stop at the Data Link layer and have to cover the Network layer issues also. New features have their cost of implementation and developments should be balanced with the key properties of the SpaceWire – compactness, economy and cost-efficient implementation in chip area and power consumption. These problems are to be investigated in further developments of the SpaceWire standard family, in SpaceFibre and SpaceWire 2 developments.

4 REFERENCES

i Tanenbaum A. Computer networks. 4th edition, Prentice Hall, 2003.
ii Michelli G., Benini L., Networks on chips. Elsevier, 2006.
iii SpaceFibre CODEC functional Specification. University of Dundee, 2007.
iv Infiniband Architecture Specification. Release 1.2 (final). 2004. Infiniband TA.
v RapidIO Interconnect Specification. Rev 1.3. RapidIO trade association. 2006.
vi Fibre Channel physical and signaling interface (FC-PH) rev 3.4. 1994.

QUALITY OF SERVICE REQUIREMENTS FOR A HIGHER LAYER PROTOCOL OVER SPACEWIRE TO SUPPORT SPACECRAFT OPERATIONS

Session: SpaceWire Networks and Protocols

Short Paper

Takahiro Yamada

JAXA/ISAS, 3-1-1 Yoshinodai, Sagamihara, 229-8510, JAPAN

E-mail: tyamada@pub.isas.jaxa.jp

ABSTRACT

SpaceWire [1] has been used on many spacecraft as a means of high-rate data transfer between onboard components. It has the capability of moving data from an onboard node to another onboard node and the capability of specifying the route to be traversed in the network. However, it does not have the capability to provide different classes of quality of service (QoS) for different types of data flows. To operate spacecraft onboard components, several types of data need to be transferred between onboard nodes and each of these data types has different QoS requirements. This paper describes the QoS classes that need to be provided by a higher-layer protocol to support spacecraft operations, and proposes simple solutions to implement them.

1 INTRODUCTION

SpaceWire [1] has been used on many spacecraft as a means of high-rate data transfer between onboard components. It has the capability of moving data from an onboard node to another onboard node and the capability of specifying the route to be traversed in the network. However, it does not have the capability to provide different classes of quality of service (QoS) for different types of data flows. To operate spacecraft onboard components, several types of data need to be transferred between onboard nodes and each of these data types has different QoS requirements. If different types of data can be transferred in a single network, we can minimize the implementation efforts and maximize the efficiency of the network. Therefore, it is desirable for most spacecraft applications of SpaceWire if there is a higher-layer protocol that runs over SpaceWire and provides necessary classes of QoS for different types of data.

There are three important measures of QoS for spacecraft applications: latency, reliability, and volume. For example, data used to monitor and control onboard components in real-time need to be transmitted over the network within a limited latency, while data related to science or mission products are not so sensitive to latency. Data used to monitor and control intelligent components need to be transferred reliably, and their losses should be detected immediately. Data generated periodically do not have to be protected against losses so severely. Data used to monitor and control components have small volumes, while data related to science or mission products have large volumes.

This paper describes the QoS classes that need to be provided by a higher-layer protocol to support spacecraft operations in section 2, and proposes simple solutions that implement the QoS classes in section 3.

2 QUALITY OF SERVICE (QOS) REQUIREMENTS

2.1 DATA TYPES USED FOR SPACECRAFT OPERATIONS

To operate spacecraft onboard components, several types of data need to be transferred between onboard nodes. There are three major types of data that are used on most spacecraft: data used for monitoring and controlling onboard components, data used for mission production, and data used for maintaining onboard components.

To monitor and control onboard components, commands are sent from a computer (it may be the central computer of the spacecraft, the attitude and orbit control computer or a payload/mission computer) to components, and telemetry is sent back form the components to the computer. There may be several different types of commands and telemetry depending on the types of the components monitored and controlled. If the component monitored and controlled is an intelligent one having a processor, commands are used to start/stop programs and change settings of the programs. The component returns high-level messages or reports about the status of the execution of the programs. If the component monitored and controlled is a non-intelligent device without a processor, commands are used to control the device directly in real-time. The device returns the status of its various parts (called housekeeping data) periodically. Intelligent components may also generate housekeeping data periodically. The computer also distributes clock data to components to inform them of the value of the master clock of the spacecraft periodically.

Science instruments generate science data that are the products of the mission. For example, cameras generate images. To maintain onboard components, especially those with processors, it is sometimes necessary to upload and download memory data to and from the components. Memory data include computer programs and large tables such as star catalogues.

Table 2-1 summarises the types of data discussed in this subsection.

Table 2-1 Types of Data used for Spacecraft Operations

Data Types		
Data used for monitoring and controlling components	Commands	For intelligent control
		For real-time control
	Clock	
	Housekeeping data	
	Reports	
Data used for mission production	Science data	
Data used for maintaining components	Memory data	

2.2 QoS Requirements for Data Types

Each of the data types described in 2.1 has latency, reliability and volume requirements. Clock data and data used for monitoring and controlling non-intelligent devices (commands for real-time control and housekeeping data) have a high latency requirement (that is, they have to be delivered to the destinations within a short time), a low reliability requirement (that is, small losses of data can be tolerated), and a low volume requirement. Data used for monitoring and controlling intelligent components (commands for intelligent control and reports) have a high latency requirement, a high reliability requirement (that is, losses of data cannot be tolerated), and a low volume requirement. Science and memory data have a low latency requirement, a high reliability requirement, and a high volume requirement.

Table 2-2 summarises the QoS requirements for each data type.

Table 2-2 QoS Requirements for Each Data Type

Data Types		Latency Reqs.	Reliability Reqs.	Volume Reqs.
Commands	intelligent control	High (Asynchronous)	High	Low
	real-time control	High (Periodic)	Low	Low
Clock		High (Periodic)	Low	Low
Housekeeping data		High (Periodic)	Low	Low
Reports		High (Asynchronous)	High	Low
Science data		Low (Asynchronous)	High	High
Memory data		Low (Asynchronous)	High	High

3 Proposed Solutions

3.1 Latency Control

Some of the data types require that data have to be delivered to the destination within a limited amount of time. Assigning dedicated time slots to these data types is a simple way of guaranteeing latency requirements for these data types. SpaceWire [1] has a mechanism of distributing 64 time codes and these time codes can be used to define time slots. A dedicated time slot (or a set of dedicated time slots) should be assigned to each data type (or a set of data types having similar latency requirements). Out of 64 time slots, a small number of time slots will be assigned to the data types with high latency requirements and the other time slots to the other data types.

To control the data traffic in each slot, there must be a master node for each time slot. If the network can be divided into multiple sub-networks each of which does not share network resources with the other sub-networks, there can be a master in each of the sub-networks. The master can either (1) send data to or receive data from the other nodes in the slot or (2) determine which node should send data to which node in the slot. For the master to be able to determine which node should send data to which

node, the master should poll data transfer requests form the other nodes and signal its decision in the same slot or in another slot assigned to such usage.

For monitor and control purposes, the node that monitors and controls the other nodes will be the master of the slot and it sends commands to and collects telemetry from the other nodes. For transferring science and memory data, the master can either (1) send or receive data or (2) determine the source-destination pair.

Since the data types with high latency requirements have low volume requirements, and vice versa, the same mechanism can be used to control volume requirements.

3.2 Reliability Control

For data types with low reliability requirements, data should be transmitted only once without retransmission because the receiver can wait for the next data without needing to receive the missing data. For data types with high reliability requirements, retransmission of missing data is necessary either through the same path or through a redundant path. For both types of data, it may be desirable to have a mechanism to let the sender know whether or not the receiver has received the data sent by the sender.

Reception acknowledgement and retransmission control should be implemented by a protocol at a layer higher than the layer that controls the slots. If the Remote Memory Access Protocol (RAMP) [2] is used, only the retransmission control should be implemented on top of RMAP because RMAP has the capability for reception acknowledgement. In this case, RMAP should be used on top of the slot control layer.

4 Conclusion

This paper has shown the QoS classes that need to be provided by a higher-layer protocol to support spacecraft operations, and proposed simple solutions to implement them.

5 References

1. European Cooperation for Space Standardization (ECSS), “Space engineering - SpaceWire - Links, nodes, routers and networks,” ECSS- E- ST- 50- 12C, January 2003.

2. European Cooperation for Space Standardization (ECSS), “Space engineering - SpaceWire - Remote memory access protocol,” ECSS- E- ST- 50- 52C, February 2010.

REAL-TIME SIGNALLING IN NETWORKED EMBEDDED SYSTEMS

Session: Networks and Protocols

Short Paper

Liudmila Koblyakova, Yuriy Sheynin, Dmitry Raszhivin

St.Petersburg University of Aerospace Instrumentation

67 B.Morskaya st., 190 000, St.Petersburg, Russia

E-mail: luda_o@rambler.ru, sheynin@aanet.ru, dmitry.raszhivin@guap.ru

ABSTRACT

The SpaceWire standard includes Time Codes that were designed for implementation of time distribution services. Using time codes the Time Access Service was developed. It provides a consistent application interface to a local time source that is maintained to be synchronised to the onboard time source master. Distributed Interrupt mechanism has been proposed for next SpaceWire standard release. Interrupt codes and Interrupt_Acknowledge codes are low-latency signalling codes and their distribution does not depend on data flow that makes it useful for real-time distributed systems interconnections. The distributed interrupts service provides real-time signalling for applications in distributed architectures with SpaceWire interconnections. The received from SpaceWire network Interrupt codes would be transmitted to user applications as real time signals with the standard POSIX Real time signals mechanism. Described services were developed for Linux OS with patches for running in soft real time.

1 SERVICES ARCHITECTURE

Linux offers embedded designers an inherently modular operating system that can be easily scaled down to compact configurations suitable for embedded designs. Plus, Linux is the fastest growing server operating system and is rapidly moving into embedded applications.

For chips manufactured by ELVEES with built-in SpaceWire channels software has been developed to work with in the OS Linux environment:

- *Drivers* for SpaceWire channel controllers, which allow to use of SpaceWire channels (links) as regular network devices. Each channel is represented by its network interface with an IP address; so all TCP/IP applications that use BSD POSIX sockets API would work over SpaceWire interconnections without any change.

- The *Time Access Service* (TAS), that provides applications with a consistent interface to a local time source that is source that is maintained to be synchronised to the onboard time source master.. The time values provided by this service might typically be used by applications to schedule some operations, such as the acquisition of an image or to time stamp locally generated telemetry data.

- *Distributed Interrupts Service* (DIS) is a service for real-time signalling with SpaceWire distributed interrupts. Its software interface for applications is the standard POSIX real-time signals interface.

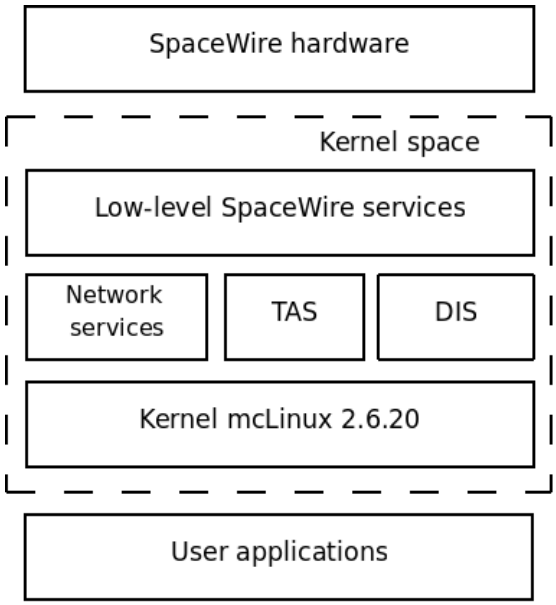


Figure 1: OS Linux architecture with SpaceWire services

2 NETWORK SERVICES

Each SpaceWire channel is represented by the Network Services that work over Low-level SpaceWire services (driver included) as a regular Linux network device with its own IP address; it supports data transmission over TCP/IP. User applications are provided with the standard POSIX socket interface, so a lot of network applications can be used over a SpaceWire interconnection without any change: http, ftp, telnet clients and servers, and a wide range of standard utilities for network configuration and diagnostics, as ifconfig, route, ping, nuttcp, etc.

3 DISTRIBUTED INTERRUPTS SERVICE

Interrupt-Code represents a system signal request. It is issued by a node link that will be considered as the source node for this interrupt (Interrupt Source). The Interrupt-Code is broadcasted to find an Interrupt Handler node. It is distributed over the network to all other nodes. An Interrupt-Code should be accepted for handling in some node of the SpaceWire network, which will be called the Interrupt Handler. The host of the node is supposed to implement some interrupt processing routine. One of 32 interrupt request signals (interrupt source identifiers) could be identified by the Interrupt-Code.

Interrupt_Acknowledge-Code represents a confirmation that the Interrupt-Code has reached some Interrupt Handler and has been accepted by it for processing. The Interrupt Handler node should send an Interrupt_Acknowledge-Code with the same five-bit interrupt source identifier as in the accepted Interrupt Code.

A signal is a limited form of inter-process communication used in POSIX-compliant operating systems (Linux included). Essentially it is an asynchronous notification sent to a process in order to notify it of an event that has occurred. When a signal is sent to a process, the operating system interrupts the normal flow of program execution. Execution can be interrupted during any non-atomic instruction. If the process has been previously registered as the signal handler its routine is executed.

The PASC Real-time System Services Working Group (SSWG-RT) has developed a series of standards that amend IEEE Std 1003.1-1990 and the profile standard (IEEE Std 1003.13-1998). The Real-time amendments to IEEE Std 1003.1-1990 is IEEE Std 1003.1b-1993 Real-time Extension. According to this standard, Linux support 32 real-time signals, ranging from SIGRTMIN to SIGRTMAX that can be used for application-defined purposes.

The Distributed Interrupts Service (DIS) uses a real-time signal to inform user applications about the interrupt or exception that has been set somewhere in the distributed system. Applications have to register at the DIS service and define the interrupt handler to receive the particular real-time signal.

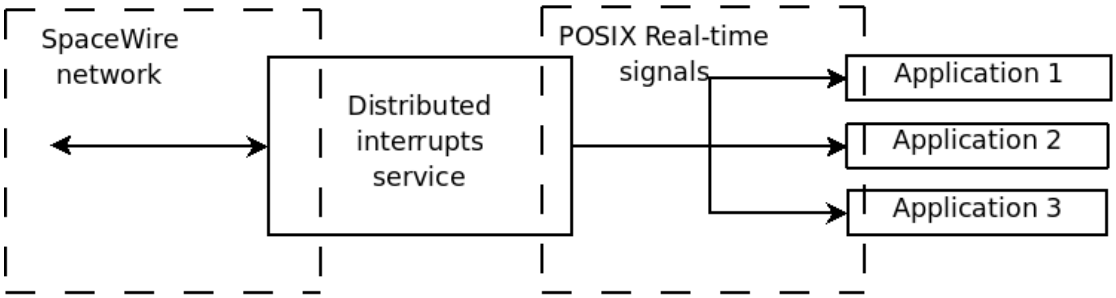


Figure 2: Distributed interrupts service architecture

4 TIME ACCESS SERVICE

The CCSDS (The Consultative Committee for Space Data System) develops standards for space systems. It proposed a draft standard SOIS version CCSDS 872.0-R-0.3, which defines the requirements for the network subsystem; it is specified in the form of services over the network.

One of the proposed services is time access service. This service allows many hosts to work with the same time. It supports synchronous and time scheduled execution of programs, which is important for onboard real-time systems.

The SOIS Time Access Service provides applications with a consistent application interface to the local time source that is maintained to be synchronised to the onboard time source master.. The time values provided by this service might typically be used

by the application to schedule some operations, such as the acquisition of an image or to time stamp locally generated telemetry data.

The SpaceWire standard has Time Codes – a tool that could be used for time distribution service implementation, though it does not suggest a ready-maid mechanism for local times synchronization in distributed systems.

In our research we have reviewed existing algorithms of time distribution, designed and realized the unified time service according to the developed algorithm. It corresponds to the standard CCSDS SIOS «Time access service», and uses SpaceWire times codes for time marks distribution.

A typical architectural scenario is shown at the Figure 3. The onboard time system architecture consists of local and master onboard time sources implemented in hardware.

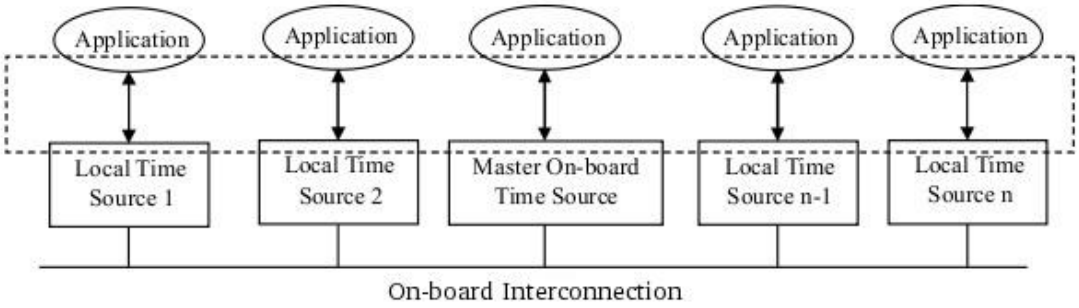


Figure 3: Typical Onboard Time System Architecture

REFERENCES

1. ECSS-E-50-12A “SpaceWire - Links, nodes, routers and networks”, European Cooperation for Space Standardization (ECSS), 2003, 124 p.

2. Sheynin Yu., Gorbatchev S., Onishchenko L., “Real-Time Signalling in SpaceWire Networks”. International SpaceWire Conference. Conference Proceedings. Space Technology Centre, University of Dundee, Dundee, 2007. ISBN: 978-0-9557196-0-8..

3. Onishchenko L., Eganyan A., Lavrovskaya I. Distributed interrupts mechanism verification and investigation by modelling on SDL and SystemC. International SpaceWire Conference, Nara 2008. Conference Proceedings. ISBN: 978-0-9557196-1-5

4. Corbet J., Rubini A., Kroah-Hartman G. Linux Device Drivers, Third Edition - O'Reilly Media, Inc., 2007.

Networks and Protocols 2

SPACEWIRE-D PROTOTYPING

Session: SpaceWire Networks and Protocols

Short Paper

Albert Ferrer, Steve Parkes

Space Technology Centre, School of Computing, University of Dundee, Dundee, UK

E-mail: aferrer@computing.dundee.ac.uk

ABSTRACT

SpaceWire-D is a new protocol developed for SpaceWire to provide deterministic delivery and offer guarantees in latency and bandwidth. This paper presents the main design drivers of SpaceWire-D, the key concepts involved, and the results of software prototyping using flight qualified SpaceWire components.

1 INTRODUCTION

SpaceWire [1] provides a versatile network architecture for onboard data-handling using switches and bi-directional serial links. It delivers the high throughput required for payload data with low implementation cost. However, it does not provide guarantees in the packet latency due to network congestion. Besides, the use of wormhole switching increases the worst case latency of packets that use shared links on the way to their destination.

SpaceWire-D [2,3] is a protocol that provides guarantees in latency and throughput by ensuring deterministic packet delivery. A Time Division Multiplexing (TDM) technique allows delivering data within predetermined time constraints. TDM is implemented using SpaceWire time-code characters sent periodically to determine the time-slots. A suitable network schedule determines when each node can send data, imposing that there is never congestion in the network. Without congestion, throughput and latency are deterministic and can be set by the user via scheduling. This is in contrast with other techniques that attempt to only mitigate network congestion and rely on network simulations to obtain throughput and latency figures.

Therefore, TDM allows assigning, independently, the worse case packet latency for command and control operations and the minimal throughput for payload data. It overcomes the traditional conflict between these two network metrics. This generic approach was also proposed for SpaceWire-RT [4], a protocol that targets reliability, in addition to timeliness issues.

The other main characteristic of SpaceWire-D is the utilization of RMAP protocol [5] to encapsulate the user data into SpaceWire packets. RMAP protocol provides a convenient way to read and write to remote memory address space using SpaceWire and is being proposed for the operation of the Plug and Play protocol. Therefore, SpaceWire-D provides deterministic delivery to these basic operations with high efficiency and low cost, without limiting further possibilities with optional functions and upper layer protocols.

2 DESIGN DRIVERS

SpaceWire-D was designed to be simple and effective. High performance requiring high complexity was avoided. Functionalities that are not required to achieve deterministic behaviour do not belong to the protocol core but instead are optionally implemented by upper level protocols. Existing SpaceWire technologies and protocols are used to take advantage of available devices and services. Finally, some flexibility is sacrificed in benefit of a low cost solution to most user cases.

3 SCHEDULING

SpaceWire defines at link level a high-priority low-latency character that provides a tick signal and an associated code that is broadcasted to all the network. Called time-code, it is a natural choice for distributing the synchronization signal that determines the current timeslot of the network.

Transaction

Most avionic systems use command and response transactions requiring a bidirectional communication. The requirements on latency and bandwidth apply to the whole transaction, not to the individual command/response packets. Therefore, the scheduling refers to transactions, and the reservation of bidirectional links, not unidirectional paths to the destination. This usually improves network usage as timeslots of fixed length are not wasted in small command packets.

Schedule table

During a specific timeslot, one or multiple nodes are allowed to initiate a single transaction, following a network schedule table. Multiple concurrent initiator nodes are allowed providing that they do not use any of the same SpaceWire links in the network. To enforce that, each initiator node may implement a local schedule vector that determines for each slot which destinations, represented by logical addresses, are valid. Note that multiple destination logical addresses can represent the same destination node but may indicate the use of different paths or different subunits within the destination node.

This destination list could be empty or could indicate that any destination is valid. If multiple destinations are provided, the node initiates a transaction with the first destination in the list with a pending transaction request. This simple priority mechanism allows to guarantee certain bandwidth and latency for the first destination in the list without loosing the bandwidth allocated when there is no pending transaction request for this destination. Besides, it provides more flexibility that the scheduling implemented for SpaceWire-RT protocol, which only allowed one destination node per timeslot.

An example schedule table is illustrated in Figure 1. It schematically represents a typical application for on board data handling. A mass memory unit is reading data from each instrument and writing data to a telemetry system, while a control processor is controlling instruments and stores housekeeping information in the Mass Memory. Therefore, the control processor unit is a initiator node, the instruments (addresses 40,41,42) and the telemetry system (address 60) are target nodes, and the mass memory (address 50) is an initiator and a target.

Time-slot	0	1	2	3	...	63
Control Processor Targets	41, 42, 50	42, 40, 50	40, 41, 50	41, 42, 50		40, 41, 50
Mass Memory Targets	40, 60	41, 60	42, 60	40, 60		42, 60

Figure 1: Example of a Schedule Table.

In this example table, the control processor only writes data to the Mass Memory when it does not have to issue control commands to the instruments, which have tighter latency requirements. A new command can be sent to any of the instruments in less than two timeslots (i.e. 100µs latency if a slot last 50µs).

4 TRANSACTION LAYER

User data is encapsulated in the data field of RMAP packets. RMAP protocol provides read and write operations on remote memory addresses with optional acknowledgement for write operations. RMAP targets are usually implemented in hardware and should execute RMAP operations within a few microseconds, excluding the reception or sending time. Even in case of error all SpaceWire data characters should be consumed at the destination so a SpaceWire link never gets blocked. This is the case of most of available RMAP implementations.

The maximum data length of a RMAP packet is limited by the protocol (i.e. 512 bytes). Bigger user data units can be accommodated by using consecutive slots or by implementing a segmentation layer.

5 FDIR FUNCTIONS

Fault detection is provided using the optional acknowledge feature of RMAP and the SpaceWire link layer error detection. This covers link errors, router and node interface failures, and all system errors covered by the RMAP protocol. Synchronization errors due to system clock failure or missing/invalid time-codes are detected using a local clock synchronized with the period of time-code arrival (timeslot period).

Upon error detection, recovery functions such as retrial mechanisms or redundancy switching are left to upper layer protocols or to the application. This reduces the complexity of the protocol and allows the user to use the best method adequate to a particular scenario.

6 PLUG AND PLAY (PNP) SUPPORT

SpaceWire-D supports plug and play efficiently, by using the same mechanism, the RMAP protocol, for its operation. New nodes attached to the network are detected by the network manager, as a result of a change in the status of the link, which the new node is attached. The network manager is responsible for the SpaceWire-D related configuration of the new node.

Alternatively, new initiator nodes can also notify the network manager of its presence, but only when the timeslot number is zero. New initiators only operates once its local clock is synchronized with the period of the time-codes received and the network manager logical address is present in the attached router configuration space. Besides, initiators can only perform read and write RMAP operations with a maximum of four

bytes. The maximum number of devices that can be connected simultaneously in a already configured network depends on the maximum data length.

If no timeslots are received and the initiator node is configured as a potential network manager then network discovery algorithms can asynchronously discover and configure the network.

7 PROTOCOL STACK

The protocol stack for SpaceWire when using SpaceWire-D is illustrated in Figure 2. Extra functionalities not directly provided by SpaceWire-D are the Packet Transfer Protocol (PTP), the segmentation function and the Retry/Redundancy layer.

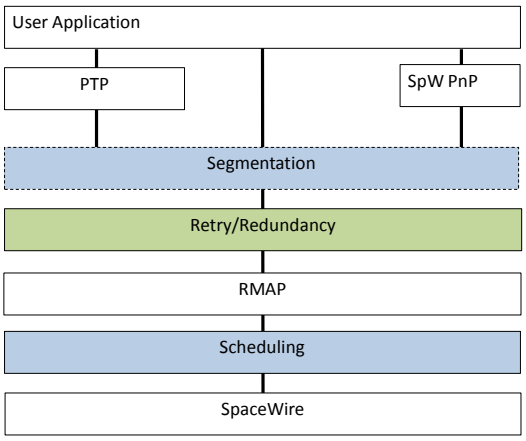


Figure 2: SpaceWire protocol stack using SpaceWire-D

The segmentation function is only required when the user data units are bigger than the maximum SpaceWire-D data length, and a schedule with consecutive slots is not considered. It operates by creating multiple RMAP transactions that use the maximum data length except the last one. The RMAP address is incremented by the data length value for each transaction. Optionally, one byte of the transaction field of RMAP can be used to indicate the user data unit sequence.

The optional retry/and redundant layer provides recovery mechanisms when a network error occurs, i.e. an RMAP acknowledge is missing. This can be check at the beginning of the next timeslot or after a arbitrary timeout has elapsed. This later case is more complex and requires the use of one byte of the transaction field of RMAP to keep track of the transaction number.

The Protocol Transfer Protocol provides the functionality required to send user messages to another node using packet buffers. Both a push and a pull type of packet transfer capability could be provided using RMAP writes or reads respectively. The transaction field of RMAP is used to identify which packets belong to a user message or user data unit. This layer can provide notifications/interruptions that a new message is available or has been sent, and the size of the message. It can also notify that one or more messages have been processed at the destination. This provides a kind of

application acknowledge and an end to end flow control mechanism that can be useful for pipelined data processing.

8 SOFTWARE PROTOTYPES

SpaceWire-D have been prototyped in software using the LEON processor of the Remote Terminal Controller (RTC, AT7913E) [6]. The essential SpaceWire-D functions and the optional segmentation function have been successfully implemented. A local clock synchronized with the time-code period triggers the sending of data with less than five microseconds accuracy. The CPU usage is low when using a data length of 512 bytes. Thanks to the use of the RMAP hardware support of the RTC, only the acknowledge packets have to be processed by software.

The user application interface is based on the configuration of local channels that define a logical address and the RMAP transaction configuration. The schedule is programmed with a list of valid channel identifiers for each time-slot. This identifier is used to confirm that an RMAP transaction has been executed without errors.

9 CONCLUSIONS

SpaceWire-D provides efficient deterministic data delivery over SpaceWire using RMAP transactions. This allows to meet latency and bandwidth requirements of the onboard network at design time. The protocol provides read/write remote memory functions with error detection capabilities. It also provides the foundations to support Plug and play, higher reliability and message transfer services.

SpaceWire-D has been prototyped on space qualified ASICs using software implementations with low CPU usage. Hardware implementations will benefit from existing RMAP components.

10 REFERENCES

1. ECSS, “SpaceWire – Links, nodes, routers and networks”, ECSS-E-ST-50-12C, July 2008, available from http://www.ecss.nl.

2. S. Parkes and A. Ferrer-Florit, “SpaceWire-D Deterministic Control and Data Delivery Over SpaceWire Networks”, ESA Contract No. 220774-07-NL/LvH, University of Dundee, April 2010, available from http://spacewire.esa.int/WG/SpaceWire/

3. S.Parkes, A. Ferrer “SpaceWire-D: Deterministic Data Delivery with SpaceWire” International SpaceWire Conference, St Petersburg, Russia, June 2010.

4. A. Ferrer-Florit, “Unified communication infrastructure for small satellites”, International Astronautical Congress, October 2009 (IAC-09.B4.6A.1)

5. ECSS, “SpaceWire - Remote memory access protocol” ECSS-E-ST-50-52C, 5 February 2010, available from http://www.ecss.nl.

6. ESA/Saab, “SpaceWire Remote Terminal Controller”, http://spacewire.esa.int/content/Devices/RTC.php .

THE ADAPTATION AND IMPLEMENTATION OF SPACEWIRE-RT FOR THE MARC PROJECT

Session: SpaceWire Networks and Protocols

Short Paper

Stuart Fowell, Patricia Lopez-Cueva

SciSys UK Ltd, Clothier Road, Bristol, BS4 5SS, U

Alan Senior

Systems Engineering and Assessment (SEA) Ltd, Somerset, BA11 6TA, UK

Omar Emam

EADS Astrium, Stevenage, Hertfordshire, SG1 2AS, UK

Wahida Gasti

ESA/ESTEC, 2200 AG Noordwijk ZH, The Netherlands

E-mail: stuart.fowell@scisys.co.uk, patricia.lopez-cueva@scisys.co.uk, alan.senior@sea.co.uk, omar.emam@astrium.eads.net, wahida.gasti@esa.int

ABSTRACT

This paper describes the application and implementation of the SpaceWire-RT protocol in the MARC project: a practical scenario, utilising representative flight hardware and next-generation network architectures. A relevant subset of the protocol was selected and implemented entirely in software and was adapted to communicate with hardware not specifically designed for a SpaceWire-RT system. Additionally, the context of a representative flight software system raised issues such as synchronisation which were not considered by SpaceWire-RT. The paper closes by summarising the lessons for timely and reliable use of SpaceWire that can be drawn from this detailed project, considering the complete communications stack from subnetwork to application interface.

1 OVERVIEW OF THE MARC PROJECT AND THE APPLICATION OF SPACEWIRE-RT

The Modular Architecture for Robust Computing (MARC) [1] is an ESA GSTP mini-project being undertaken by SciSys, Astrium UK and SEA. MARC is developing a decentralised onboard computer [2] using a SpaceWire network on a backplane and SOIS [3] services as a communication backbone with a hierarchical FDIR mechanism. The MARC demonstrator architecture is illustrated in Figure 1.

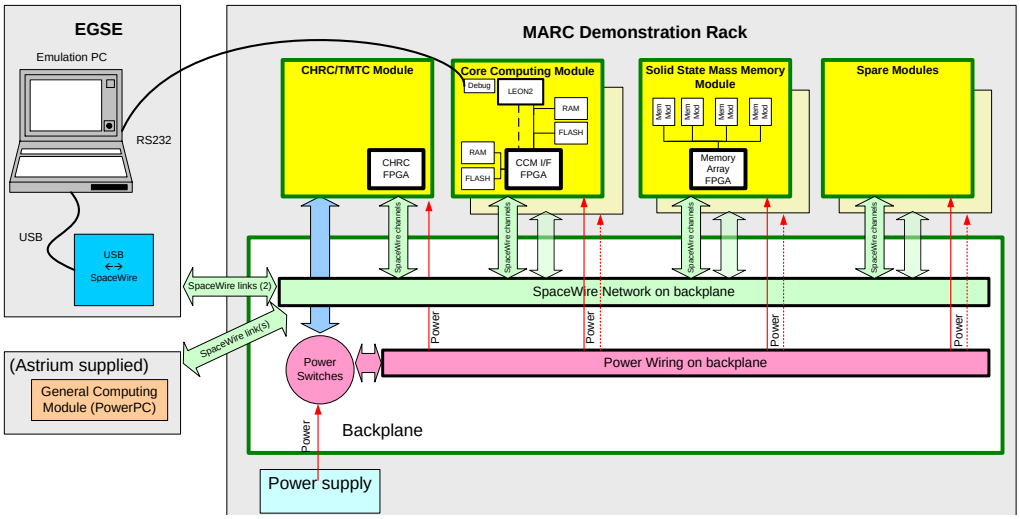


Figure 1. MARC Demonstrator Architecture

The hardware architecture is closely coupled to the software aims of the Generic Fault-tolerant Software Architecture using SOIS (GenFAS) software framework, developed by SciSys. This provides a PUS-based Data Handling Services, communication functions using SOIS, FDIR management and a software deployment and upgrade mechanism. The GenFAS software architecture is illustrated in Figure 2.

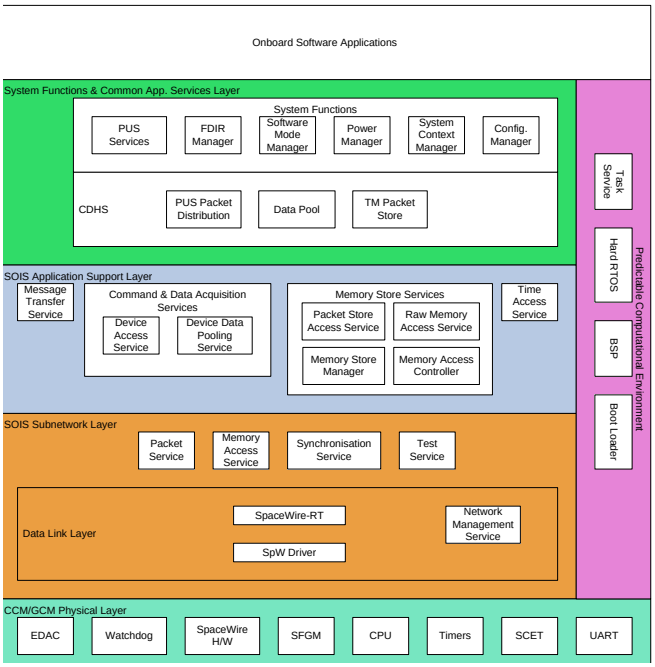


Figure 2. GenFAS Software Architecture

A crucial part of the SOIS software stack for the MARC project is the provision of a suitable SpaceWire service guaranteeing timely delivery of data. To achieve this, SciSys applied the proposed SpaceWire-RT protocol [4]. The protocol aims to ensure timeliness by utilising time-codes to divide the available network bandwidth in pre-allocated slots and specifies facilities for reliability and redundancy management.

2 ADAPTATION AND IMPLEMENTATION OF SPACEWIRE-RT

2.1 GOALS AND CONSTRAINTS

It was a key goal of MARC to implement SpaceWire-RT to provide the required SOIS Quality-of-Service (QoS) classes for communication across the MARC SpaceWire network, making use of the scheduled SpaceWire-RT configuration. This then is to be used as a basis for assessing the implementation and use of the SpaceWire-RT draft specification.

The implementation was constrained to be made only in software, with the ESA RMAP [5] and SpaceWire Codec IP cores being employed on the MARC Core Computing Modules (CCMs). It was anticipated that implementing SpaceWire-RT protocols, including flow control, in software would have performance implications.

A second constraint was that communication must be supported with legacy RMAP-based nodes. This has two implications; communication between CCMs and legacy nodes must be by using RMAP and that the legacy nodes would have no knowledge of the SpaceWire-RT imposed network schedule. As a consequence, the CCMs must be able to send and receive RMAP packets with no SpaceWire-RT protocol encapsulation and the SpaceWire network communication must be managed such that the legacy nodes never asynchronously transmit SpaceWire packets, i.e. it is only in a manner synchronised to the SpaceWire network such that it can be taken into account when determining the SpaceWire-RT schedule.

2.2 SELECTION OF SPACEWIRE-RT FEATURES AND ADAPTATIONS

Based on an assessment of typical information flows and associated QoS required by functions in MARC (function chains, FDIR etc), the following features of SpaceWire-RT were selected for implementing:

- Basic, Best-Effort, Assured and Reserved QoS (Guaranteed not required).
- No redundancy (handled at a system level by switching SpaceWire network plane in a coordinated manner by the FDIR applications).
- No group adaptive routing (incompatible with SOIS QoS, not required for redundancy and no information flow required multiple, parallel SpaceWire links that it provides).
- No prioritisation of Reserved QoS traffic (not required for information flows, subsequently removed from SOIS).
- No “opportunistic” allocation of unused, reserved time-slots (overcomplicates scheduling analysis and unnecessary).
- Simplified API based on maximum user packet sizes (optimising copying).

A number of the requirements and constraints of MARC were not met by the SpaceWire-RT specification and so the following extensions and adaptations were required:

- Addition of Raw channels (Best-Effort or Reserved QoS, no SpaceWire-RT encapsulation) and Raw time-slots (scheduling of raw channels) – used for communication with legacy RMAP or PTP-based nodes.

- RMAP packets limited so that a transfer fits within the duration of a time-slot. RMAP reply packets assumed to be received on channel number + 1 from that used for command packet.
- Extended flow control bit fields (to support larger buffers).
- No kill mechanism implemented (not fully defined and not able to implement in software with existing IP cores).
- Addition of a controlled configuration mechanism for nodes and routers (initialisation and any subsequent re-configurations by each node while SpaceWire-RT schedule suspended).
- Integration with SOIS Synchronisation Service (dual use of SpaceWire time-codes for time distribution and synchronisation of SpaceWire-RT time slots).

2.3 IMPLEMENTATION OF SPACEWIRE-RT

Figure 3 is an illustration of the implementation of SpaceWire-RT.

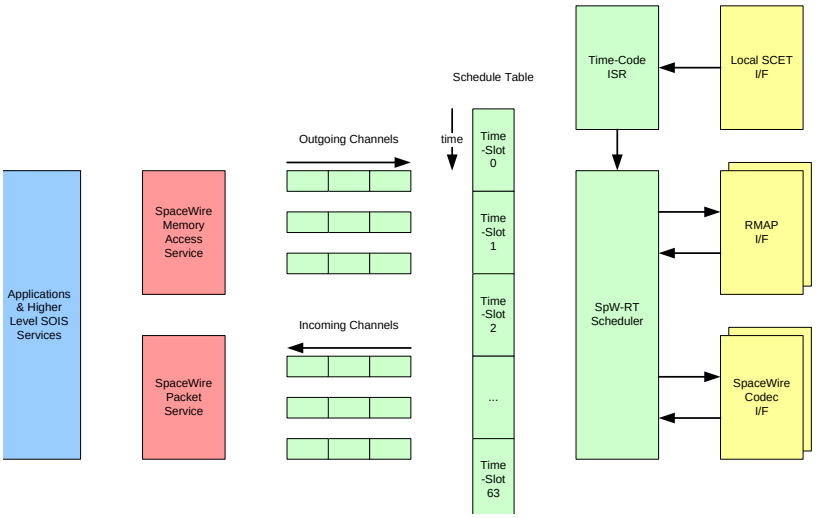


Figure 3. SpaceWire-RT Implementation Block Diagram

The channel interface from the SpaceWire-RT specification was preserved as much as possible, so as to minimise the effort required in any future port to a hardware-based implementation. Special consideration was made for handling of RMAP and the use of RMAP IP Core as a hardware accelerator. For non-RMAP packets, the RMAP IP Core's bypass mechanism was used, with additional hardware support functionality for calculating CRCs. Of course, because SpaceWire-RT was implemented in software on a single processor; send and receive, multiple ports, etc. had to be implemented in a serial algorithm.

The SOIS Memory Access Service mapped straight onto RMAP. For the SOIS Packet Service, the CCSDS Packet Transfer Protcool (PTP) [6] was employed with different information flows requiring both raw PTP packets and encapsulated in SpaceWire-RT.

Finally, tuning of the rate of sending time-codes was required, trading-off between resolution of time distribution and maximum rate at which time slots could be handled.

3 LESSONS LEARNT ON THE USE OF SPACEWIRE-RT

SpaceWire-RT is both very feature-rich and yet still being prototyped. To make practical use in an onboard environment to flight standards requires selection of only features appropriate to the development (reducing complexity and cost of validation) and to make assumptions, extensions and adaptations to overcome the current status of its (incomplete) specification. Standardisation of an appropriate subset and recommendations into its use in conjunction with existing and forthcoming SpaceWire and SOIS standards will simplify its deployment.

Use of SpaceWire-RT requires careful management of the complex configuration of channels and time-slots across each node in the SpaceWire network. As a SpaceWire network is inherently more complex than, say, a MIL-STD-1553B bus, offline analysis tools are strongly recommended to analyse communication scenarios, based on actual information flows and how they map down to packets on the network, and to automatically generate the resulting configuration data. Part of this is addressed by the analysis tool being produced by EADS Astrium as part of the MARC project [7].

Clearly a hardware-based implementation would be more performant, handling send/receive in parallel and multiple ports. However, the decision on the hardware/software split should also take into account the adaptability of software.

4 CONCLUSIONS

An adapted subset of SpaceWire-RT has been successfully implemented in software using hardware-support functions, integrated with SOIS services and is being used in the MARC demonstrator to provide managed timely communications across a SpaceWire network in a decentralised onboard computer.

5 REFERENCES

1. "Generic FT Software Architecture using SOIS for MARC", ESTEC Contract Number 20863/07/NL/LvH.

2. "Advanced Robust Processing Architecture 'ARPA' for Modular Architecture for Robust Computing 'MARC'", ESTEC Contract Number 21034/07/NL/LvH.

3. CCSDS, "Spacecraft Onboard Interface Services – Informational Report", CCSDS 850.0-G-1, Green Book, Issue 1.0, June 2007

4. Parkes and Florit, "SpaceNet – SpaceWire-RT Initial Protocol Definition", SpW-RT WP3-200.1, Draft A Issue 2.1, 30 October 2008.

5. "Space Engineering – SpaceWire – Remote Memory Access Protocol", ECSS-E-ST-50-52C, 5 February 2010.

6. "Space Engineering – SpaceWire – CCSDS Packet Transfer Protocol", ECSS-E-ST-50-53C, 5 February 2010.

7. Emam *et al*, "A Software Analysis Tool Supporting FDIR Management for Systems with SpaceWire Networks – MARC Project", Proceedings from the 3rd International SpaceWire Conference, St. Petersburg, Russia, 2010.

SPACEWIRE NETWORK SIMULATOR

Session: SpaceWire Networks and Protocols

Short Paper

Artur Eganyan, Liudmila Koblyakova, Elena Suvorova
Saint-Petersburg University of Aerospace Instrumentation. 67, B. Morskaya, Saint-Petersburg, Russia
E-mail: artfla@rambler.ru, luda_o@rambler.ru, wildcat15@yandex.ru

1 INTRODUCTION

The simulation is an important task for device design, distributed systems building and network protocols development. Therefore, previously we have developed the SpWNM [1] intended for simulation of SpaceWire networks. In the course of further development on this project we have created SpaceWire Network Simulator (SpaNSim).

The SpaNSim is intended for designing, modeling and analyzing the SpaceWire networks of any topologies, and contains basic models of terminal node, routing switch and link. The SpaNSim has all features described in [1]: it allows designing and simulation of SpaceWire networks; implements wormhole routing, time flow and distributed interrupts mechanisms, generation and transmission of data packets; estimates workload of channels and devices; provides graphical networks design with MS Visio.

Unlike the SpWNM, SpaNSim:

- allows the user to create new device models and to write applications for them;
- allows to add new types of devices operating on different transmission standards and to connect these devices with SpaceWire networks;
- displays state of any device's internal element: buffers content, register values and so on;
- contains a lot of modules for network analyzing: for example, one can see an in-depth information for each packet; can analyze in detail the network at level of bits, symbols, packets or particular events;
- displays all bits transferred through the channel and shows distorted bits;
- allows to describe the networks not only graphically, but with usual C++ projects, for example, in Visual Studio.

The SpaNSim is implemented in C++ and based on Qt and SystemC.

2 NETWORK DESIGN AND DEVICE SETTING

With SpaNSim you can graphically design the networks, using MS Visio. Also, the SpaNSim provides you to configure parameters and state of any device's internal elements in interactive mode: buffers, registers, memory, clocks, and so on. It can be done during the network design in MS Visio or during the modeling. For example, you can check or change device's buffers content after the modeling performed for some period.

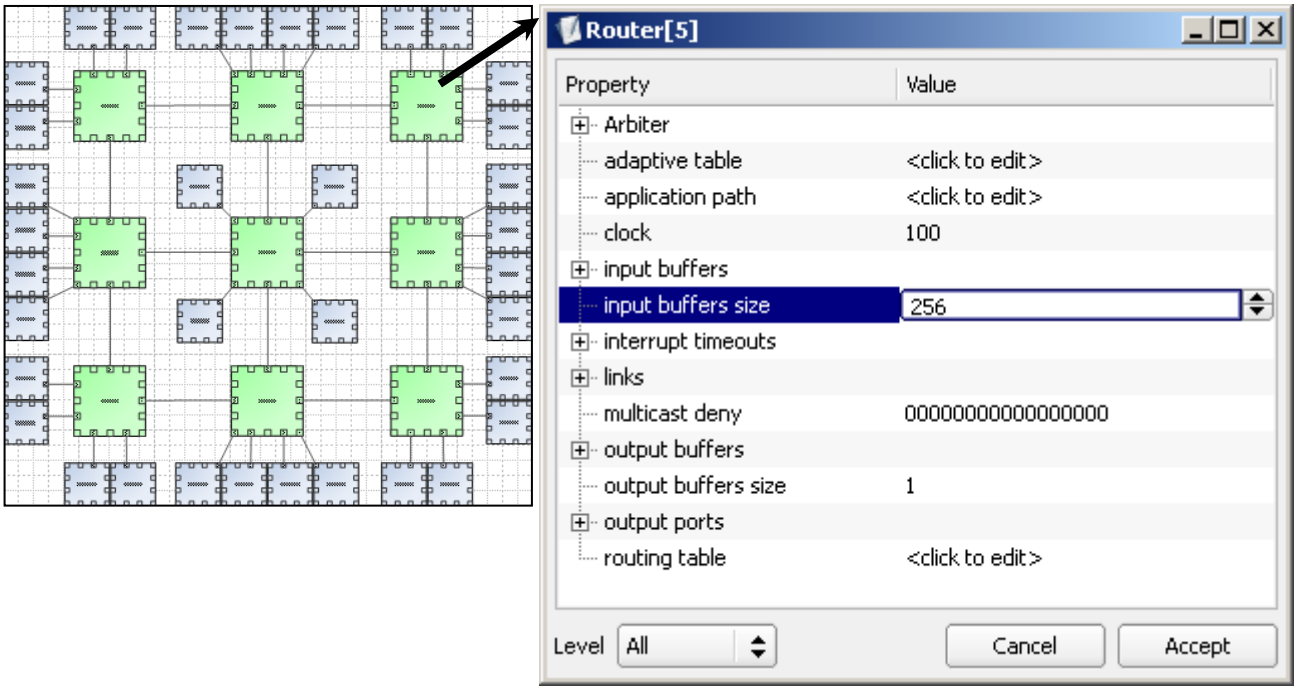


Fig.1. Network design and device setting in MS Visio.

3 CREATING NEW DEVICES

The SpaNSim provides you to create new device models. The new model can be implemented on basis of the existing one or without usage of existing models. For example, new device can be created by changing links of the terminal node or changing the routing switch's arbitration or buffering scheme, channels number, and so on. For fast development, SpaNSim provides a set of ready-to-use functional units, so you can create a new device, connecting them with one another and setting their parameters.

Also you can write applications for the devices. For example, you can create an application operating inside the terminal node and sending/receiving data packets and control codes. For this, you should not know the node in details, you will only implement the application algorithm and use an interface between the application and node. Using this interface, your application will send or receive data packets, change transfer rate and so on. Each application will be presented as a independent program unit written in C++ or SystemC and attached to the node, so you can write various applications and estimate how they operate in the SpaceWire network.

In SpaNSim it is possible to add new types of devices. For example, one can implement a bridge binding the SpaceWire network with another ones: MIL-STD-1553, CAN, RS422-485 and so on. Hereby, networks of devices operating on different transmission standards can be implemented and modeled together.

4 ANALYZING SIMULATION RESULTS

The SpaNSim includes a lot of modules for analyzing modeling results. Using them the user can: estimate minimum, average and maximum propagation time of data packets, interrupts and time-codes; see information about any packet or control code transfer; see symbols flow transferred through the channels; see bit flows and find out bits distorted under the noise; get information about all occured timeouts. Also, the SpaNSim automatically builds charts displaying minimum, average and maximum propagation time for data packets and control codes.

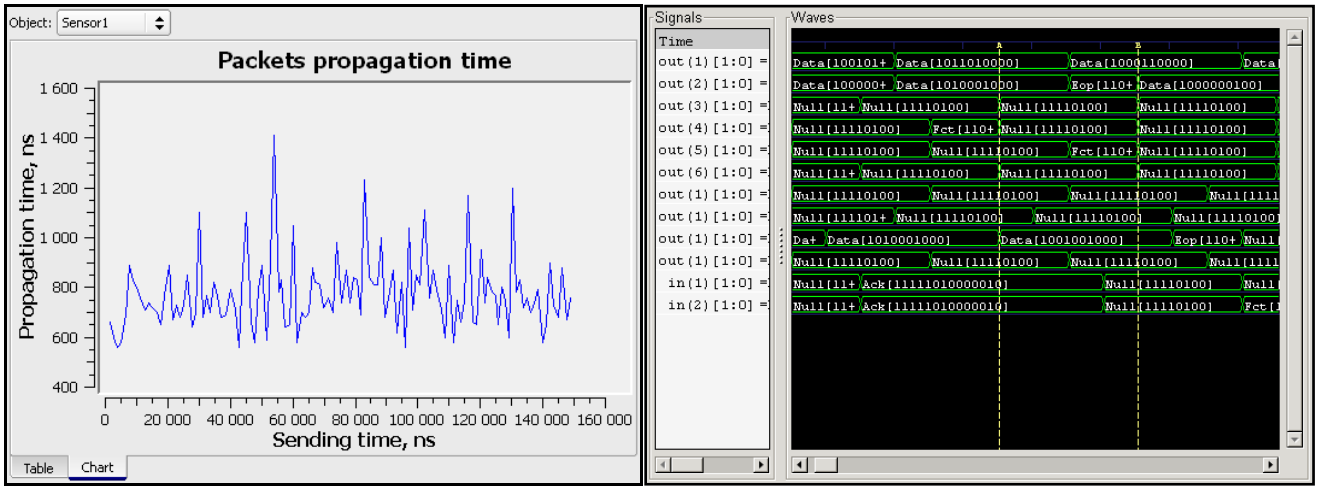


Fig.2. Analysis of data packets propagation time and displaying bits transferred through the SpaceWire channels.

Under development of new devices, the user can specify events and graphical dependencies he is interested in, and they will be appended to the results.

5 SPANSIM AND OPNET MODELER

The OPNET Modeler is one of the most popular software for network design and simulation. Therefore we have compared it with the SpaNSim relatively to SpaceWire networks simulation and analyzing.

In the default configuration of OPNET Modeler, there are no SpaceWire models ready to use. To simulate SpaceWire networks with OPNET, firstly you should have developed at least all basic devices: terminal node, routing switch, link and channel. For this, it is needed to write a lot of code and build finite-state machines implementing sending and receiving of data packets and control codes, supporting wormhole routing, adaptive group routing and so on. If you are going to transfer packets symbol-by-symbol (for example, to implement a wormhole routing), in OPNET each symbol should be defined as a packet itself. Bit-level transferring and analyzing (e.g., in channel with a noise) will be more difficult for implementation in the Modeler. In the SpaNSim, all of this is already implemented.

In the table below you can see comparison between some general characterictics and features of the OPNET Modeler and SpaNSim.

What is compared	SpaNSim	OPNET Modeler, v. 14.0
Network graphical design	Yes	Yes
Device graphical design	No	Yes
Setting of devices and their components	Yes	Yes
Displaying device's buffers and registers content	Yes	No by default

Programming languages used for device development	C++, SystemC	Primarily C. It is possible to use C++ and scripts.
Modeling core performance, in events per second	Approximately equal	
Information units	Symbols inside devices and bits inside the channels	Packets formatted by the user
Modeling results provided by default	Analysis of each packet and control codes. Analysis of minimum, average and maximum propagation times. Displaying transferred and distorted bits. Displaying a content of internal buffers and registers.	Animation for packets transfer. Minimum, average and maximum propagation times. Propagation time and a route for each packet.
Modeling results defined by the user	Any additional charts and text messages	Any additional charts and text messages
Modeling the networks of different protocols	Yes	Yes

By default, the OPNET Modeler does not support bit-level transfer and analysis, and does not provide detailed analysis of device buffers and registers. In SpaNSim, all these features are implemented.

In the Modeler, if you want to write an application for some device, you will usually change the device code. In the SpaNSim, devices provide convenient interfaces for applications, so you should not think about the device code, you only implement the application's algorithm.

Also, a new device in the SpaNSim is developed on the high-level language (C++ or SystemC) and with some ready-to-use modules. During programming, the user has a direct access to all device parameters, and he can use intuitive data types like «simulation time», «symbol», «packet», «routing table», and so on. In the OPNET Modeler, any device component is described with finite-state machine, and the user puts some C code in the states. Access to the device parameters is not as clear as it in the SpaNSim. So, in the OPNET Modeler it can be difficult to design a very detailed device model.

6 CONCLUSIONS

The SpaNSim provides graphical design, simulation and analysis of SpaceWire distributed systems. It has a set of ready-to-use devices and allows you to create new device models operating on SpaceWire or different standards. As a result of simulation, it is possible to estimate a wide range of characteristics required for building the distributed systems, and to define systems' parameters.

REFERENCES

1. Elena Suvorova, Liudmila Onishchenko, Artur Eganyan, "SpaceWire Network Functional Model", Session "SpaceWire networks and protocols", International SpaceWire Conference, Dundee 2007.

FAULT TOLERANT SPACEWIRE ROUTING TOPOLOGY AND PROTOCOL

Session: SpaceWire Networks and Protocols

Short Paper

Muhammad Fayyaz and Tanya Vladimirova

Surrey Space Centre, Department of Electronic Engineering, University of Surrey, Guildford, GU2 7XH, UK

E-mail: fayyazrafiq@hotmail.com, t.vladimirova@surrey.ac.uk,

ABSTRACT

This paper is concerned with a fault tolerant routing topology and a protocol for SpaceWire networks on board spacecraft. A fault tolerant routing topology is proposed, which is comprised of cross strapped central SpaceWire router core. The central core is cross strapped in such a way that no single faulty node results in getting down the whole system. In addition, a routing protocol is proposed for the support of dynamic updating of the routing table. The main characteristic of the routing protocol is that any faulty node is immediately isolated from the rest of the system.

1 INTRODUCTION

Space systems require a high degree of reliability, which can be achieved via redundancy. SpaceWire is a full duplex, serial, point to point data communication standard. It is a high speed serial bus with speed limit up to 400Mbp/s and uses low voltage differential signalling (LVDS) for the transmission of data. The SpaceWire standard describes the protocol with respect to physical, signal, character, exchange, packet and network levels [1].

Fault tolerant routing provides redundant paths for routing of packets during the failure of the primary routing node. Group adaptive routing supports link redundancy only, but in case of a failure in any routing node the whole system gets down, which is not acceptable for space missions. In order to enhance the reliability of SpaceWire networks, a fault tolerant routing topology is proposed, which is comprised of cross strapped central SpaceWire router core. The central core is cross strapped in such a way that no single faulty node results in getting down the whole system. In addition, a routing protocol is proposed, which supports dynamic updating of the routing table. The main feature of the routing protocol is that any faulty node is immediately isolated from the rest of the system.

2 FAULT TOLERANT ROUTING TOPOLOGY

A fault tolerant topology scheme using SpaceWire is presented in Figure 1, where each node is considered as dual redundant, i.e. primary and redundant. Table 1 shows the failure mode effects analysis (FMEA) for the fault tolerant routing topology.

Table 1. Failure Mode Effects Analysis of the Fault Tolerant Routing Topology

S. No.	Possible Failure	Effect on the System	Remedy
1	Node to router Link Failure	Loss of Primary Link	Switch to similar node redundant link
2	Node Failure	Loss of Primary Node	Switch to redundant node
3	Router to Router Link Failure	Loss of Primary Link	Switch to redundant link
4	Router Failure	Loss of Primary Router	Switch to redundant router

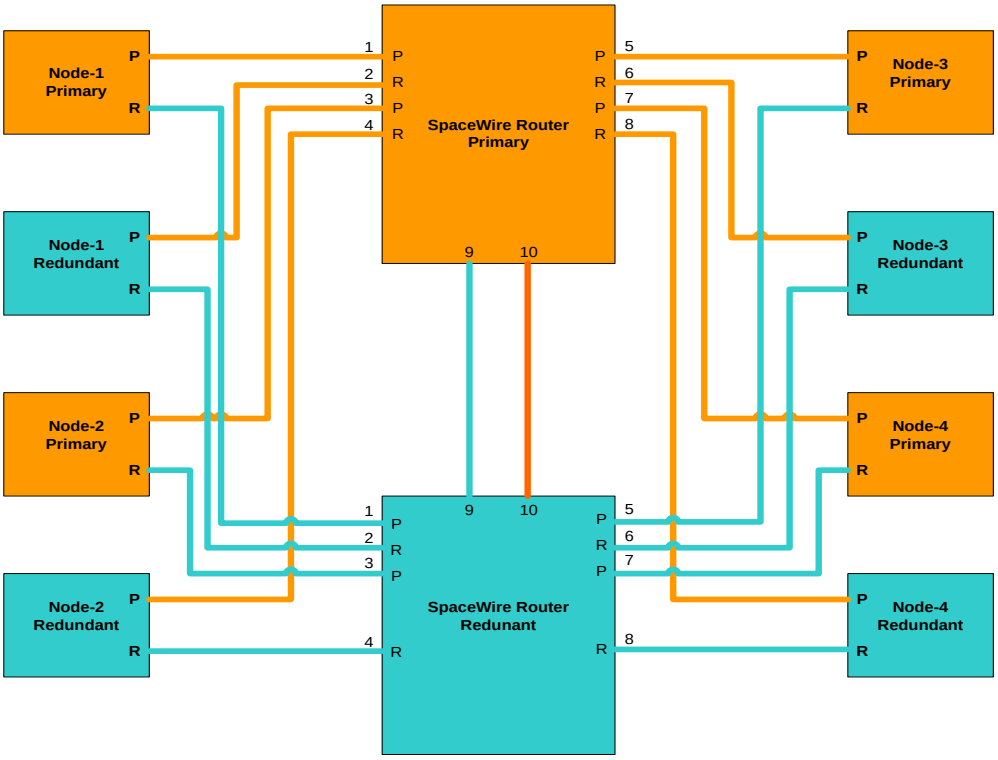


Figure 1. Fault Tolerant Routing Topology

The nodes can be hot redundant or cold redundant. Failure of any hot or cold redundant nodes will not affect the overall system performance. In Figure 2, if a link from node-1 primary to the SpaceWire router primary fails, it will switch to the node-1 redundant link and will route the packets via the redundant router. Similarly if node-1 primary fails, then it will switch to redundant node-1 and will route the packets via the primary router. If the link between the routers fails it will switch to a redundant link. In case of router failure the links between routers and node to router can be used for the exchange of routing information or for the transfer of data packets.

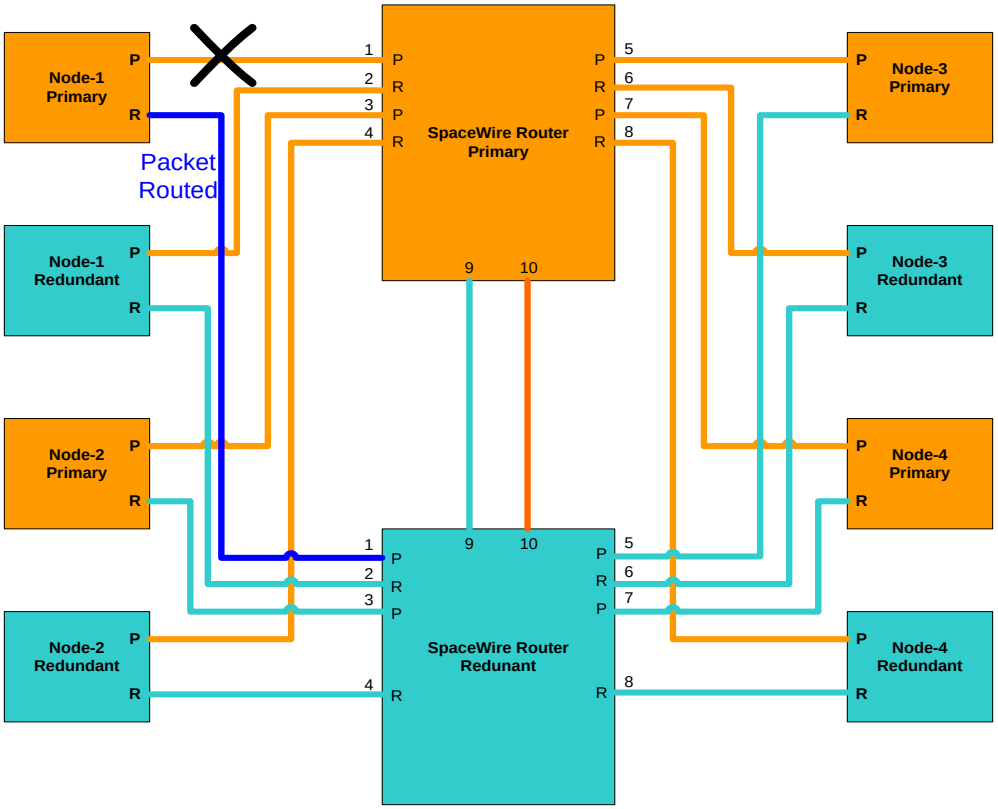


Figure 2. Failure of a Node Link

3 FAULT TOLERANT ROUTING PROTOCOL

In order to support the fault tolerant routing topology presented in the previous section, a fault tolerant routing protocol is required. Table 2 shows the additional message exchange for the fault tolerant routing topology.

Table 2. Fault Tolerant Routing Protocol Keep Alive Messages

Node/Router	Node-1 (P)	Node-1 (R)	Router (P)	Router (R)
Node-1 (P)	No Msg Exchange	No Msg Exchange	Node-1(P) says to Router(P): I am active	Node-1(P) says to Router(P): I am active
Node-1 (R)	No Msg Exchange	No Msg Exchange	Node-1(R) says to Router(R): I am active	Node-1(R) says to Router(R): I am active
Router (P)	Router (P) says to Node-1(P): I am active, Not Busy, Node-1(R) active, Router(R) active.	Router (P) says to Node-1(R): I am active, Not Busy, Node-1(P) active, Router(R) active.	No Msg Exchange	Router (P) says to Router(R): I am active, following links available or busy or faulty.
Router (R)	Router (R) says to Node-1(P): I am active, Not Busy, Node-1(R) active, Router (P) active.	Router (R) says to Node-1(R): I am active, Not Busy, Node-1(P) active, Router (P) active.	Router (R) says to Router (P): I am active, following links available or busy or faulty.	No Msg Exchange

There are mainly three types of messages: node to router, router to node (reply message) and router to router. The formats of these messages are designed in such a way that there is no additional overhead on the network performance. For the support of these packets at network layer, a source logical address is used. Table 2 presents the partial routing table of the primary router. In case of a link failure between the primary router and the primary node-1 no keep-alive messages are exchanged between them and the router considers this node as a dead node. It also informs the redundant node with logical address 10 that the primary node is dead. Now the redundant node starts sending packets to others nodes via the primary router. If the primary node-1 is up again, then the router updates the entry status from down to active and restores the communication with the primary node-1.

Table 3. Partial Routing Table for the Primary Router

Source Logical Address	Destination Logical Address	Out port	Priority	Status
5	15	3,9,10	High	Down
10			Low	Active
5	20	4,9.10	High	Down
10			Low	Active
5	25	5,9,10	High	Down
10			Low	Active
5	30	6,9,10	High	Down
10			Low	Active
5	35	7,9,10	High	Down
10			Low	Active
5	40	8,9,10	High	Down
10			Low	Active

4 CONCLUSIONS

The presented fault tolerant routing topology and its protocol fulfil the requirement of space systems. The fault tolerant architecture considered here is for two routers and eight nodes but it can be extended for any numbers of nodes. Future work will involve the implementation of this conceptual design.

5 REFERENCES

1. European Corporation for Space Standardization, “Space Engineering SpaceWire-Links, nodes, routers and networks”, 31 July 2008.

A SOFTWARE ANALYSIS TOOL SUPPORTING FDIR MANAGEMENT FOR SYSTEMS WITH SPACE WIRE NETWORKS –MARC PROJECT

Session: Space Wire Missions and Applications

Short Paper

Omar Emam, Allan Whittaker, Simon Lentin, and Tony Jorden

EADS Astrium, Stevenage, Hertfordshire, SG1 2AS, UK

Wahida Gasti

ESA/ESTEC, 2200 AG Noordwijk ZH, The Netherlands

E-mail: omar.emam@astrium.eads.net , allan.whittaker@astrium.eads.net, simon.lentin@astrium.eads.net, tony.jorden@astrium.eads.net, wahida.gasti@esa.int

ABSTRACT

A custom-designed off-line software tool has been developed for analysing the Space Wire "SpW" network performance of a fault-tolerant system, based on the "Modular Architecture for Robust Computing"(MARC) concept, after the occurrence of a Fault Detection, Isolation and Recovery (FDIR) event. The tool also generates the FDIR and re-configuration tables required by the system's onboard FDIR management software. This tool is referred to as the MARC 'FDIR Analysis Tool' which has been devised to support the design and implementation of MARC- based systems. The MARC concept is outlined, and the role of the "FDIR Analysis Tool" is described in this paper.

1 INTRODUCTION

The MARC project [1] aims at developing a fault-tolerant decentralised onboard computing architecture, using a high-reliability SpW network as its communication backbone. This is an UK-GSTP/ESA-funded project undertaken by Astrium UK, SciSys, and SEA. The FDIR Analysis tool has been developed by Astrium, as part of its role in specifying and developing the MARC FDIR strategy and related architecture. The tool will be used extensively in Astrium's validation of the MARC concepts and designs, using the demonstrator hardware [2] and software, towards the end of 2010.

The complexity of advanced networking systems, with their multitude of parameters to be taken into account, makes it almost impossible to design an optimum network solution simply by manual inspection. The use of some form of system analysis tool has therefore become essential to support the design of complex computing networks where there are multiple data exchange paths, with different data traffic characteristics and constraints.

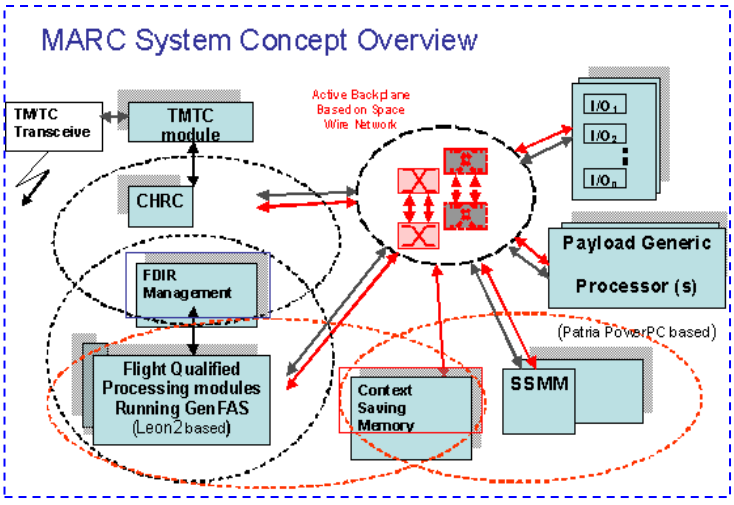
MARC is one such advanced computing network system for which the MARC 'FDIR analysis tool' was conceived, as an off-line software application that would be run at the design phase of this complex system. The tool is used to analyse the suitability of a given MARC SpW network, in terms of its throughput and latency, to meet the

system requirements. The tool is also used to enable the user to run FDIR scenarios which are intended to analyse the system performance after a fault recovery and highlight any non-conformances. The tool then generates the FDIR and re-configuration tables associated with these FDIR conditions. The resulting tables are used by Astrium's MARC 'FDIR Manager' [4] which is implemented by SciSys as part of their Generic Fault-tolerant Software Architecture "GenFAS" software framework [1] developed for MARC.

2 FDIR ANALYSIS TOOL AS PART OF MARC

In a MARC system multiple computing, memory, and Input/output (I/O) nodes (or modules) are interconnected via a high-reliability SpW network. MARC is a scalable architecture. In its simplest form it resembles a standard on-board computing system, with a set of prime and redundant nodes/modules, all connected to single pair of prime and redundant SpW routers to form a simple single-hop (single router) network. In this case it is relatively simple to ensure, by inspection, that the resulting network will meet the specified performance requirements. On the other extreme, up to 24 nodes of different types (computing, memory, and I/Os) can be connected to a number of chained routers to form a complex heterogeneous network. The problem of designing the system is made more difficult since it is possible to have multiple data paths between nodes with different packet lengths as well as different data rates and link speeds. In addition, there may be requirements on data traffic prioritisation, in that some data packets, such as command, Health-Status or Timing-Data packets have to be routed from source to destination within a restricted timing window. In this case, designing a system that guarantees the specified performance cannot be achieved without the aid of some analysis tool. Finally, the tool has to enable the user to generate the FDIR and re-configuration tables required by "FDIR manager" onboard software which is used to implement the FDIR strategy. No off-the-shelf tools have been identified which meets these specific needs of a MARC system. In particular, a network analysis tool for MARC must include the capability to define the parameters for the exchange of the Health Status messages necessary for implementing the MARC FDIR strategy.

3 HOW THE TOOL WORKS



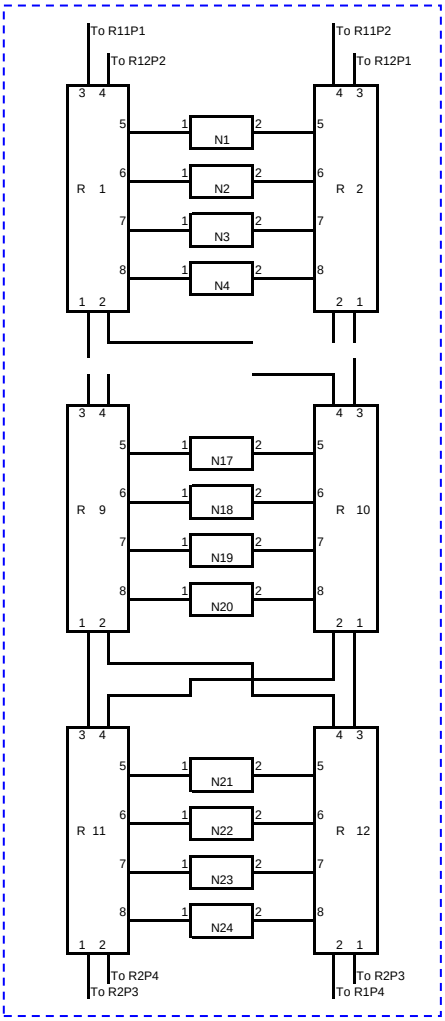
The tool takes as its input a user defined file that specifies the SpW network architecture being considered for analysis. The first stage of this analysis is to determine what all possible network connections exist between the various nodes. The tool also checks that the network conforms to the MARC standard network architecture - see following figure, as this is important for generating the FDIR related tables. Once the tool has established that any node can connect to any other node via at least two paths, the network performance analysis can then start. This analysis is used to validate the

robustness of the MARC network in terms of fault tolerance, by enabling the user to mimic failure and failure recovery in the system and identify any non-compliances in its performance after recovery. For example swapping a prime computing node on one router with a redundant node on another router and showing that the resulting network still meets the system performance requirements.

By performing the validation for all possible network failures and failure recovery conditions the tool is able to generate a set of FDIR and configuration tables that the on-board FDIR Manager software uses to indentify a fault recovery procedure. The onboard software combines the entry in the table, pointing to the current configuration, with the identifier of the diagnosed fault to generate a pointer. This allows it to identify an entry in the FDIR and configuration table that defines the configuration of the recovered system.

3.1 SPACE WIRE NETWORK PERFORMANCE ANALYSIS

The network performance analysis has two main objectives: to analyse the throughput of the network and to determine the latency of data packets passing through it. In the analysis, the tool allows for packets of different sizes and different rates, as well as two different Space Wire link frequencies (see below). In addition, the tool permits multiple data communication channels between any node and any other node, which creates a 'virtual network'. This is important since it enables the user to allocate the different data channels different priorities for a SpW-VN [5] based system, or different time slots for a SpW-RT based system [3].



In summary, the key steps for network performance analysis are:

Throughput:

- Determine which of multiple possible paths are used for a node-node connection.
- Calculate and display the percentage loading contribution from all data communication channels in a link.
- Flag if the total loading percentage of a link exceeds a defined threshold.

Latency:

- Calculate Latency for a particular path (sum of router & link contributions).
- Routers –calculate a packet's port-to-port time: worst case, all other ports are already occupied.
- Latency is assessed for each data communication channel (of a virtual network, if there are multiple paths per link).

Both the latency and the throughput takes into account that the network can be heterogeneous in terms of SpW link frequencies, being either 100MHz or 10MHz.

Latency Algorithm:
(The following values are based on using Atmel 10X SpW router):

T_{sl}	switching latency:	a constant value of 133 nanoseconds,
T_{ppl}	port to port latency:	a constant value of 547 nanoseconds
T_{bpb}	bits per byte:	a constant value of 8 bits.
T_{bpp}	bytes per packet:	32 – 1M bytes (typically 1024)
T_{bps}	bits per second	in the range of 80-70% of SpW link frequency.
T_{ppr}	ports per router:	8 ports

T_{fbd}	first byte delay:	$T_{sl} + T_{ppl}$
T_{spb}	seconds per byte:	T_{bpb} / T_{bps}
T_{spp}	seconds per packet:	$T_{spb} \times T_{bpp}$
T_{tropp}	time router occupied per port:	$T_{spp} + T_{fbd}$
T_{mtro}	max time router occupied:	$T_{tropp} \times (T_{ppr} - 2)$

3.2 MARC FDIR AND CONFIGURATION TABLES GENERATION

The operator starts the FDIR and configuration tables' generation by selecting the Generate Tables function in the Traffic Results Window. The system performs the following sequence of actions to generate FDIR tables for the pre-defined logical network:

1. Generate a bit pattern in a 64-bit word to represent the defined system 'start' configuration word, (i.e. assuming no failures), where a 1 indicates an active component and 0 represents an inactive component. When the flight system is in the corresponding configuration, the FDIR manager uses this as an "**index"** to the FDIR data it needs. For that configuration word, the router tables are then written into the Configuration Table.

2. For that configuration an element to be simulated as the 'failed' element is selected. The tool works out the new set of active elements, given the presumed failure and the known redundancies. The tool then analyses the new configuration to see whether it meets throughput & latency constraints. If not, it will write out a special configuration word defining a "safe mode" to be adopted when this faulty configuration is reached and sets a bit representing a 'non-operational flag'.

3. If the new configuration **does** meet throughput and latency constraints, the tool will write the configuration to the FDIR and configuration Tables. The sequence repeats from step 2, but with a new element chosen to be the 'failed' element. This continues until all elements in turn have taken the role of the 'failed' element.

4. When all elements for that initial start up configuration have been exhausted then a new 'start' configuration is defined, and the whole sequence is restarted from step 1. The overall process finishes when all combination of failures have been exhausted for all the start configurations. The failures should be cumulative, such that, for example, a 12 node system results in 2^{12} configurations.

4 WHAT THE TOOL LOOKS LIKE

The tool expects the system architecture to conform to that agreed within the MARC framework in that it consists of a number of different types of nodes which are interconnected via SpW links. The nodes may be processor platforms, memory modules, I/O modules, routers, etc. The user specifies the number and type of nodes as well as the interconnect between them. The tool includes a basic model of each type of node. The analysis tool has the following key user interaction stages:

4.1 THE CONFIGURATION INPUT FILE

This represents a file of network configuration data which is created before running the analysis tool. It contains text which defines what each port of each element in the physical network, is connected to. The format is:

<Router>,<Port>,<Destination>,<Link_Speed>

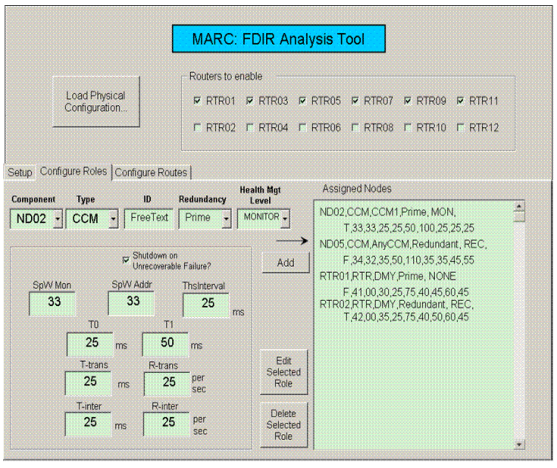
For example:
RTR01,1,RTR03,100
Meaning: Router 1, port 1 is connected to Router 3 and has a link speed of 100 MHz
ND01,1,RTR01,10
Meaning: Node 1, port 1 is connected to Router 1 and has a link speed of 10 MHz

4.2 THE PHYSICAL NETWORK WINDOW

This is the first stage in the network analysis process. Here the operator loads the "Configuration Input file" which the analysis tool then examines by performing some context checking to ensure that there were no obvious errors in the configuration file. From the GUI, the user can manually select or modify the configuration of the prime and redundant routers in the network. The analysis tool then determines the number of links (hop-numbers) between any two nodes on the network and displays it as a matrix. The user can then select an individual cell of the matrix to get information on the network links associated with that node-to-node data path.

4.3 LOGICAL NETWORK WINDOWS

This window allows the operator to assign roles (functions) to each of the nodes in the physical network, including whether the role is prime or redundant. The operator is also given the facility to define the routes between prime nodes and what level of data traffic will be carried.



4.4 TRAFFIC RESULTS WINDOW

This window allows the operator to initiate analysis of the pre-defined logical network. It is from this window that the FDIR and configuration tables' generation can be initiated.

5 CONCLUSION

A complex SpW network based computing system could not realistically be designed and implemented without the aid of some analysis tool. In the case of MARC the 'FDIR analysis tool' is an essential part of the system and is key to generating the FDIR and configuration tables that lie at the heart of the MARC FDIR strategy.

6 REFERENCES

1. Gasti et al, "Modular Architecture for Robust Computation – MARC", ISC2008 Conference. Nara-Japan, Nov 2008

2. Senior et al, "Modular Architecture for Robust Computation – MARC", ISC2007 Conference, Dundee-UK, Sept 2007

3. Fowell et al, "the Adaptation and Implementation of Space Wire-RT for the MARC Project", ISC2010 Conference, St.Petersburg-Russia , June 2010

4. Emam et al, "An FDIR Strategy based on Message Exchange Approach to Implement Autonomous FDIR Management on the MARC System", DASIA 2010 Conference, Budapest- Hungary, June 2010

5. Barry et al, "Virtual SpaceWire Networks: A Mechanism for Real-Time Applications", SpaceWire Working Group Meeting 13, ESTEC - Noordwijk - The Netherlands, Sept 2009

THE SPACEWIRE-PNP PROTOCOL IN THE SOIS PLUG-AND-PLAY ARCHITECTURE

Session: SpaceWire Networks and Protocols

Short Paper

Peter Mendham, Stuart Fowell

SciSys UK Ltd, Clothier Road, Bristol, BS4 5SS, UK

Chris Taylor

ESA/ESTEC, 2200 AG Noordwijk ZH, The Netherlands

E-mail: peter.mendham@scisys.co.uk, stuart.fowell@scisys.co.uk, chris.taylor@esa.int

ABSTRACT

The SOIS architecture has been steadily evolving for a number of years and includes an effort to incorporate the features of "plug-and-play" systems which are relevant to onboard communications. This paper reviews the role that plug-and-play can play in an onboard system architecture, and within SOIS more specifically. The SOIS plug-and-play architecture is described and the role of the subnetwork is highlighted specifically. The SpaceWire-PnP protocol is assessed against the requirements for a SOIS implementation, and is presented as an exemplar of a subnetwork plug-and-play protocol.

INTRODUCTION

The SOIS plug-and-play architecture has been steadily evolving for a number of years, with input from a range of studies and prototyping activities. Additionally, recent work has benefited from the experience of NASA with Space Plug-and-Play Avionics (SPA), clarifying the alignment of such international efforts with the SOIS architecture. As an onboard communications technology with growing popularity, SpaceWire has always been carefully considered within the SOIS plug-and-play architecture and previous work has described potential mappings of SOIS plug-and-play onto SpaceWire [1]. Concurrently, the SpaceWire-PnP protocol is being developed with the intention of providing standard mechanisms for carrying out crucial network management and configuration tasks on SpaceWire networks, including device and network topology discovery.

Section 00 begins this paper by reviewing the plug-and-play problem within the larger context of onboard system architecture, from applications to hardware. The SOIS plug-and-play architecture is the focus of this discussion, described in Section 0, where the roles that a subnetwork plug-and-play protocol must fulfil to meet the requirements of SOIS are highlighted. In Section 00, SpaceWire-PnP is applied within the SOIS architecture as an exemplar subnetwork protocol, with an examination of the

portions of the architecture that are subnetwork specific, and those that are mission specific. Section 00 concludes the paper.

ONBOARD PLUG-AND-PLAY

The SOIS Green Book [2] defines plug-and-play as "the set of automated mechanisms used to discover, learn the capabilities of, and provide access to a *device* in a spacecraft's onboard (sub-)network". One of the goals of SOIS is provide standards for such mechanisms to improve the interoperability and portability of onboard systems. An implementation of SOIS is typically software-centric, although this does not have to be the case. We will use software terminology here, beginning with onboard applications.

The focus of device access by an onboard application is the functional characteristics of that device type. For example, accesses on a sun sensor will be largely concerned with acquiring the position of the sun, whilst those on a reaction wheel will be largely concerned with commanding the wheel torque. The application is not concerned, except by necessity, with the underlying mechanism required to acquire the sun position or command the wheel torque. By abstracting the generic operations of a device type, or class, from its underlying access mechanisms, application software can be made more portable. Such a technique is known as *device virtualisation*, and forms an integral part of plug-and-play strategies on many platforms.

The presentation of a virtual device interface to an application relies on two processes: access to the device, potentially using some device-specific protocol; the adaptation of the semantics of the device access mechanisms to match the expected semantics of the virtual device. In turn, the device-specific protocol will rely on the mechanisms provided by the technology used to communicate with the device, i.e. the subnetwork. The definition of the Device-Specific Access Protocol (DSAP) and translation mechanisms for virtualisation could be implemented directly into software (or hardware). More flexibly, however, they could be described in a structured, machine-readable form such as an Electronic Data Sheet (EDS). The use of an EDS permits the automatic generation or tailoring of implementations handling the DSAP and/or virtualisation translations.

Which devices are present in the system, may be statically "hard-coded" into the communications software, or it maybe dynamically determined at run time. In the case of a static system, it may be still be necessary to verify the configuration of services against which devices are logically present on the subnetwork. This verification requires the same underlying subnetwork mechanism: to be able to discover and uniquely identify all devices on the subnetwork. If the subnetwork requires management and configuration in order to be able to access and use devices to their full potential, this must be carried out, according to some mission-specific policy. The dynamic detection of devices also raises the possibility of dynamically utilising an EDS, hosted by the device, to suitably adapt or tailor the DSAP and/or virtualisation translations associated with the device.

The subnetwork must therefore support standard mechanisms to support the access of devices, which must be supported by plug-and-play mechanisms such as device discovery and the management of subnetwork configuration.

THE SOIS PLUG-AND-PLAY ARCHITECTURE

The SOIS Plug-and-Play architecture instantiates the major features of the mechanisms described in Section 0 in explicit service interface descriptions. It is expected that these service interfaces will be implemented by suitable services, relying on the underlying hardware or driver software for a subnetwork such as SpaceWire. An example onboard software architecture implementing the SOIS plug-

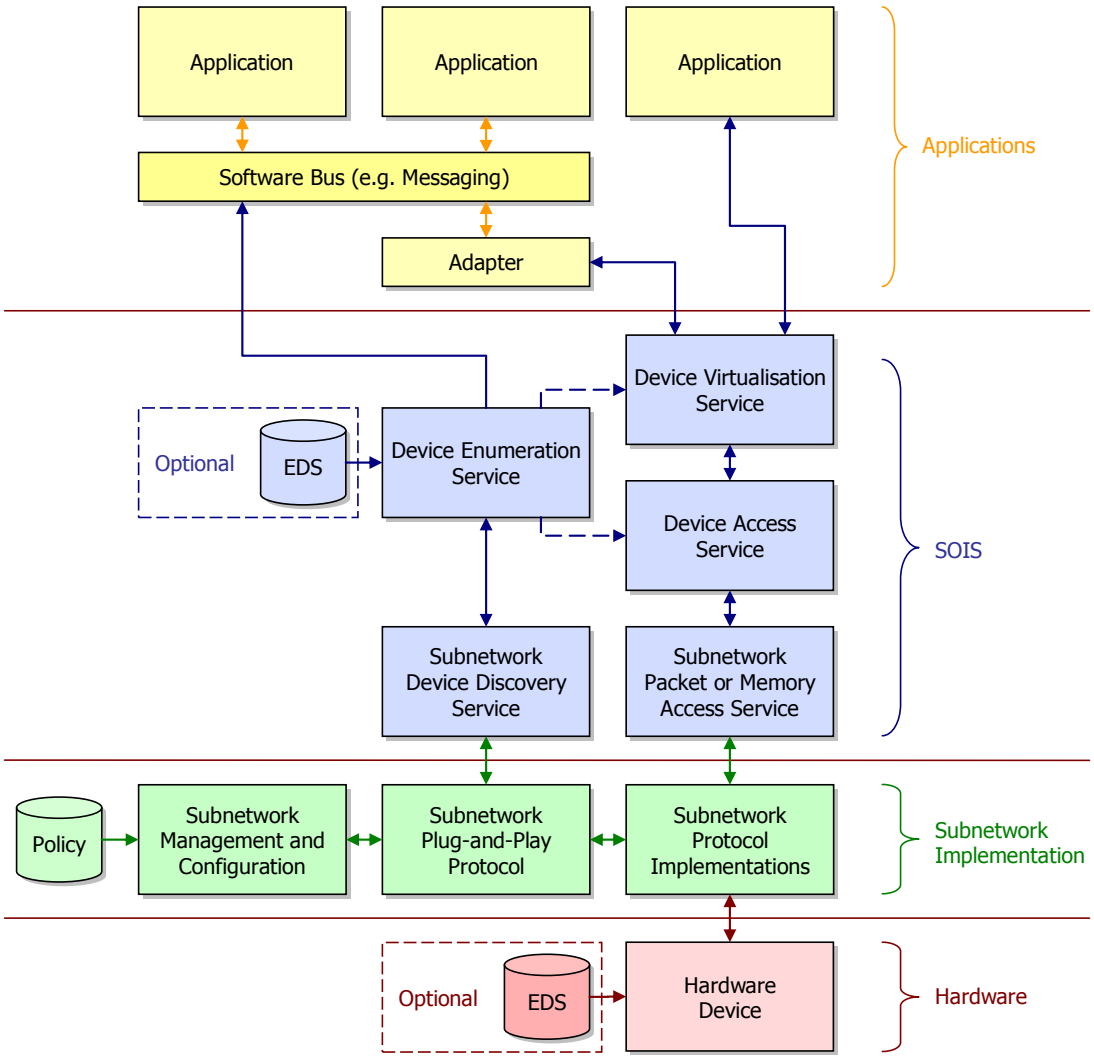


Figure 1: Onboard Software Architecture Using SOIS Plug-and-Play

The area marked as "Subnetwork Implementation" corresponds to the SpaceWire interface, and SpaceWire protocol handling. This layer is the appropriate place to handle subnetwork plug-and-play which must support: device discovery functions sufficient to support the subnetwork Device Discovery Service (DDS) and support for managing and configuring all standard subnetwork features. To permit dynamic configuration using an EDS, subnetwork plug-and-play should support a standard mechanism for querying an EDS from a device.

Whereas the plug-and-play protocol and mechanisms can be generic for a subnetwork, actual subnetwork management activities may well be specific to a mission. To reflect

this, the policy guiding the management and configuration of a subnetwork has been explicitly depicted in Figure 1.

SPACEWIRE-PNP FOR SUBNETWORK PLUG-AND-PLAY

The SpaceWire-PnP proposal [3] was initially developed by one of the authors whilst at the University of Dundee [4-7] and has continued with input from SciSys [8-9] and the SpaceWire community. The core of SpaceWire-PnP is the following services:

- Device Identification, which provides basic information about devices such as their vendor ID and how many ports they have.
- Network Management, which permits the discovery of devices and network topology.
- Link Configuration, which permits the configuration of SpaceWire links on a device.
- Router Configuration, which permits the configuration of routing features and applies only to SpaceWire routers.

Additionally, SpaceWire-PnP is extensible: so-called "capability services" provide for, amongst other things, generic data sources and sinks. The simplest source service was designed considering the requirement for hosting EDS data in a device.

As can clearly be seen, the services provided by SpaceWire-PnP meet the requirements for the SOIS plug-and-play architecture. Device Identification and Network Management services of SpaceWire-PnP provide all functions necessary for the subnetwork independent SOIS DDS, although there is an opportunity to make these correlate more closely. The Link and Router Configuration services provide for the subnetwork specific configuration and management of all SpaceWire features identified in the SpaceWire standard.

SUMMARY AND CONCLUSIONS

The SOIS plug-and-play architecture supports the application of plug-and-play principles, including virtualisation and device discovery using static and or dynamic methods. This architecture relies on device discovery features provided by the subnetwork, which must also take responsibility for management and configuration of subnetwork resources. This combination of discovery, configuration and management functions must be met by suitable subnetwork plug-and-play mechanisms, such as SpaceWire-PnP.

The flexible topology, peer-to-peer approach and distributed resources of SpaceWire are, in many ways, a superset or more general case of the characteristics provided by other communications media. The SpaceWire-PnP proposal is therefore a good model for the development of other plug-and-play protocols. Ongoing work at SciSys is prototyping key elements of SpaceWire-PnP, including the principles behind device discovery and electronic data sheet use. This work clearly indicates the potential power and flexibility of applying SpaceWire-PnP within the SOIS plug-and-play architecture.

REFERENCES

1. Fowell and Taylor, 'The SOIS Plug-and-Play Architecture and its Proposed Mapping onto SpaceWire', Proceedings of the 1st International SpaceWire Conference, Dundee, UK, 2007.

2. CCSDS, 'Spacecraft Onboard Interface Services – Informational Report', CCSDS 850.0-G-1, Green Book, Issue 1.0, 2007.

3. University of Dundee, "SpaceWire-PnP: Protocol Definition', Version 2.1, Available from the minutes of the 14th SpaceWire Working Group, http://spacewire.esa.int/WG/SpaceWire/, 2010.

4. Mendham *et al*, 'Plug-and-Play Technology for SpaceWire: Drivers and Alternatives', IAC-07-B4.7.06, Proceedings of the 58th International Astronautical Congress, Hyderabad, India, 2007.

5. Mendham *et al,* 'Interoperability and Rapid Spacecraft Development Using SpaceWire Plug-and-Play', IAC-08-B4.7.02, Proceedings of the 59th International Astronautical Congress, Glasgow, UK, 2008.

6. Mendham *et al*, 'Network Management and Configuration using RMAP', Proceedings of the 1st International SpaceWire Conference, Dundee, UK, 2007.

7. Mendham and Parkes, 'SpaceWire Plug-and-Play: A Roadmap', Proceedings of the 2nd International SpaceWire Conference, Nara, Japan, 2007.

8. Mendham *et al*, 'Standardised Sensor and Actuator Interfaces with SpaceWire-PnP', IAC-09-B4.7.02, Proceedings of the 60th International Astronautical Congress, Daejeon, South Korea, 2009.

9. Mendham, 'The SpaceWire-PnP Protocol', Available from the minutes of the 14th SpaceWire Working Group, http://spacewire.esa.int/WG/SpaceWire/, 2010.

DESIGN OF A WIRELESS LINK FOR SPACEWIRE NETWORKS

Session: SpaceWire Networks and Protocols

Short Paper

Jean R.Paul and Tanya Vladimirova
Surrey Space Centre, Department of Electronic Engineering, University of Surrey, Guildford, UK, GU2 7XH
Email: j.paul@surrrey.ac.uk, t.vladimirova@surrey.ac.uk

ABSTRACT

A large variety of new applications could be implemented in future space missions by combining the SpaceWire and IEEE 802.11 communication protocols. Converting SpaceWire packets into IEEE802.11 packets presents challenges due to bridging a protocol designed for point-to-point links with a protocol usually operating in an ad-hoc mode.

This paper presents an approach to developing a wireless interface for SpaceWire on-board networks for the purpose of inter-satellite communication in networked distributed satellite systems. The tradeoffs of a bridge design supporting SpaceWire/IEEE802.11 data transfers are discussed.

1. INTRODUCTION

Future networked distributed satellite systems (DSS) will be formed of satellites with cross links and on-board autonomy capabilities. In these systems, a spacecraft can gather information from and aggregate its resources with other spacecraft to perform tasks. In such a context, inter-satellite communication is an important feature of the satellite network [1, 2]. A distributed satellite system for space weather monitoring is proposed in [3], in which satellites are able to exchange data via an inter-satellite link (ISL) and a master node is selected to communicate with ground. In case if the master satellite fails, the network will need to reconfigure in order to select another node as master or incorporate a new member satellite.

SpaceWire is a recently developed on-board spacecraft communication protocol, which has already been deployed in a number of space applications. In contrast, the IEEE802.11 wireless standard is a mature terrestrial protocol that offers a wide range of services. Both standards have attributes supporting scheduling, flow control and buffering that can be exploited to provide a communication medium enabling high-speed fault-tolerant wireless networks. SpaceWire has a good electromagnetic compatibility (EMC) performance, is easy to implement, and supports fault-tolerance by providing redundancy to networks. The use of SpaceWire as the communication protocol for intra-spacecraft components can improve robustness and reconfigurability by adding fault-tolerance on board each spacecraft within a DSS network. These attributes make SpaceWire an attractive communication protocol on board spacecraft in distributed satellite systems [3].

This paper presents an approach to developing a wireless interface for SpaceWire on-board networks for the purpose of inter-satellite communication in future networked DSS [2, 3]. The work presented in this paper is concerned with the design of a bridge to connect a SpaceWire router to a Wireless transceiver. The bridge contains a module that is used to convert SpaceWire format into IEEE802.11 packets. A synchronization module is included to manage the flow between the router and the bridge whilst the Gaisler GRLIB memory controller [4] provides off-chip access to the SpaceWire router and modules.

2. System Overview

A high-performance computing reconfigurable system-on-a-chip (SoC) design is proposed in [3] to support data processing and inter-satellite communication on board satellites. The SoC is centred around the LEON3 processor, which is used to run software applications and AMBA2 is used as the on-chip bus system as shown in Figure 1. An IEEE802.11 transceiver intellectual property (IP) core is also included for inter-satellite communication. As IEEE802.11 based communications are asynchronous, a direct memory access (DMA) core is added to the design to control the data transfers between the memory and the wireless transceiver.

SpaceWire is a high-bandwidth and fast switching protocol in which link connection, as well as error recovery, takes 20 microseconds. The protocol is flexible in terms of network topology, there is no limitations in the packet size, and the data rate is only constrained to receiver's buffer size. In contrast, the IEEE 802.11 standard is a contention-based network protocol and, depending on the standard, the bandwidth is limited to 20 or 40 MHz. Packet size is limited to 2310 bytes and the link setup between two links in IEEE 802.11 nodes takes a minimum of 250 μs to start. In that time a SpaceWire node could send in excess of 200 data characters, when operating at 20 MHz. As a result, buffer management and storage are required to avoid bottlenecks in the SpaceWire networks while the wireless transceiver starts up a link. The DMA is able to connect AMBA ready devices to the memory controller, and the bridge interfaces with the DMA via the AMBA AHB bus.

3. Bridge Design

3.1 Link Connection and Synchronisation

The IEEE 802.11 wireless network standard, often referred to as WiFi, is defined at the Medium Access Control (MAC) and Physical (PHY) layers [5]. The IEEE802.11 physical layer can either be based on the Orthogonal Frequency Division Multiplexing (OFDM) or spread spectrum. Due to bandwidth scarcity in wireless networks, a common approach is to use a multiple access scheme to share the bandwidth of a communication link between several nodes. The MAC layer ensures that frames are delivered error free, and adds addressing information to the transmitted frames.

Inter-frame timing constraints are introduced at the IEEE 802.11 MAC layer in order to support high data rates. Before a node is allowed to initiate a transmission, it senses the channel to verify whether it is free for a predefined minimum period called Distributed Inter Frame Space (DIFS). If the channel is busy, a random back-off interval is calculated to determine the waiting time before the sending node tries to

access the channel again. This is followed by a flow control mechanism between the sending and receiving nodes. SpaceWire has a flow control for link connection as well, however once the link between two nodes is established, the data transfer rate depends only on the receiving buffer size. Typically, a SpaceWire node has a buffer able of storing 7 characters. Before a data transfer, the sender checks whether the receiver's buffer is full and data transfer will not occur until the receiver sends an authorisation to transmit to the source node.

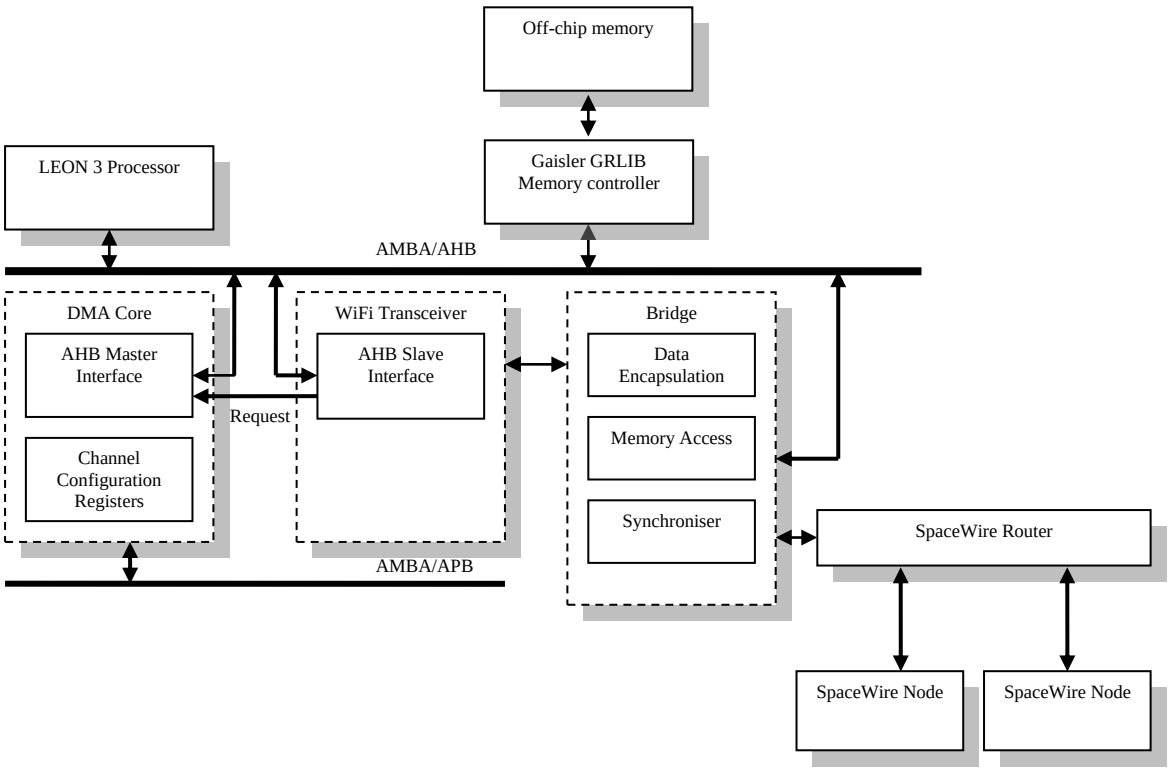


Figure 1: Integration of SpaceWire with the IEEE802.11 standard in a SoC design

In the proposed design, the bridge must keep the SpaceWire router synchronised with the WiFi transceiver. Since the data rate of the OFDM-based IEEE 802.11 nodes is set at 6 Mbps and the bandwidth is 20 MHz, for ease of implementation, the SpaceWire router is also limited to 20 MHz. The IEEE 802.11 wireless transceiver in Figure 1 is implemented as a hardware accelerator in VHDL supporting the highest data rate. Its MAC layer contains functions such as 'byte by byte' processing in both receive and transmit directions, cyclic redundancy check (CRC) generation for error detection purposes, signals to indicate successful transmissions and reception. It was observed that when the data rate is set to 6 Mbps, if a frame is received by the transceiver, the MAC layer forwards the bytes to the bridge at an interval of 350 ns and the SpaceWire destination router's buffer is never full. Even at higher data rates, the MAC would be forwarding data at the same interval which suggests that the bridge is able to efficiently transfer data between the router and a wireless transceiver operating at the highest speed for the IEEE 802.11a, g, and n standards.

3.2 Data Encapsulation

Data coming from the SpaceWire router are presented to the bridge in groups of 9 bits, in which the MSB represents the control bit, a '0' is used for data characters and a '1' is used for end of packet (EOP) and error end of packet (EEP). In order to ensure SpaceWire packets size are within the maximum 2312 bytes packet size supported by IEEE 802.11, a packet size is set in the SpaceWire network. As the wireless transceiver's MAC layer processes on a byte-by-byte to perform CRC, the SpaceWire character's control bit is stripped off in order to comply with the 8-bits input of the MAC layer. The bridge placed at the receiver re-inserts the control bit and the end of packet marker. And when the MAC layer detects an error in the packet by CRC, the MAC layer informs the bridge, which in turns sends error end of packets (EEP) characters to the SpaceWire router.

3.3. Remote Memory Access

The latency involved in establishing a link between WiFi nodes and the use of the transceiver by software applications can lead to a bottleneck in the SpaceWire network. Therefore a mechanism for data storage while the router is waiting for its turn is required. The Remote Memory Access Protocol (RMAP) is proposed to allow SpaceWire nodes to access memory directly [6]. This protocol specifies acknowledged/non-acknowledged and verified/non-verified implementation methods for read and write operations. RMAP is an application layer protocol, in which CRC is performed at both the transmitter and receiver to ensure that data is written error-free in the memory.

Functions such as CRC checking and memory access via the DMA provide the WiFi transceiver with features that present similarities to RMAP. In IEEE802.11 networks an Acknowledgement (ACK) packet is sent to the transmitting node when the destination node receives an error-free. If an ACK is not received within a duration specified by the transceiver's acknowledgement time, the transmitting node will assume that an error or collision occurred.

The long propagation delay in space is a major cause for decreased throughput. This would be further exacerbated by having to exchange a packet between spacecraft to ensure the receiving router is running and, once the SpaceWire is deemed connected, data transfer can be initiated. This may not be an advantageous flow control process, as the latency involved in WiFi link connections would be in the order of hundreds time higher in comparison to the latency of SpaceWire networks. Also, a transmitting SpaceWire node can send erroneous data in case of transient or intermittent faults [7], which would result in an inefficient use of the ISL. In the transmission of SpaceWire packets via the ISL, a link is instead assumed to be connected when a router is ready to send data to another router via the IEEE 802.11 wireless transceiver. This approach presents some distinctive advantages where even if the data originated from the SpaceWire network is error-free, they are still susceptible to the effect of the wireless communications channel effects. The transmission of a WiFi packet of 1500 bytes has a duration of 2 milliseconds and if during that period the router is still disconnected, the link can be considered to be down. The CRC performed by the WiFi MAC layer is able to detect failures due to channel impairment as well as errors in SpaceWire packets. Thus when the source node receives an Ack, channel effects are mitigated and the SpaceWire packet is error-free.

The bridge transfers data from SpaceWire nodes to the off-chip memory via the DMA. When an error is detected, the MAC layer informs the bridge, which in turns appends an EEP to the SpaceWire packets. As opposed to RMAP, the CRC checking is done at a lower level than the application layer, this in turn reduces the latency and the complexity involved in implementing direct memory access into memory-mapped SpaceWire nodes. The DMA is designed with the flexibility of allowing the off-chip memory access to be either hard-coded or set in a C program.

In essence, the bridge is capable of spoofing the end to end SpaceWire link connections. The resulting drawback is a drop in throughput in instances where a router is not connected.

4. CONCLUSIONS

This paper outlines a novel bridge design to connect SpaceWire routers with an IEEE802.11 transceiver. The similarities between the two standards are exploited and the design of a bridge capable of converting from the one standard format to the other is presented. To the best of the authors' knowledge a bridge design permitting the translation of SpaceWire packets into WiFi packets has not been previously proposed. The bridge is also able to control information flow and has a mechanism to provide memory access to SpaceWire routers. The bridge is incorporated into a high-performance SoC, which provides a communication platform enabling spacecraft with SpaceWire networks to communicate via inter-satellite links based on the IEEE 802.11 wireless network standard.

5. REFERENCES

1. K. Sidibeh and T. Vladimirova. "A Fault Tolerant High Speed Network for Inter Satellite Links", Proceedings of 8th Military and Aerospace Applications of Programmable Logic Devices and Technologies International Conference (MAPLD'2005), Washington DC, US, NASA ,September 7-9, 2005, P-144,.
2. T.Vladimirova and K.Sidibeh. "WLAN for Earth Observation Satellite Formations in LEO", Proceedings of 2008 ECSIS Symposium on Bio-inspired, Learning, and Intelligent Systems for Security (BLISS'08), Edinburgh, IEEE Computer Society Press, , August 4-6 2008,pp. 119-124.
3. T. Vladimirova, C. P. Bridges, J. R. Paul, S. A. Malik and M.N.Sweeting. "Space-Based Wireless Sensor Networks: Design Issues", Proceedings of 2010 IEEE Aerospace Conference, Big Sky, MT, 6-13 March 2010.
4. GRLIB IP Library User's Manual, http://gaisler.com/products/grlib/grlib.pdf(last accessed 01/04/2010)
5. IEEE 802.11 Wireless Local Area Networks, http://ieee802.org/11/
6. S.M. Parkes, S. Mills, and C. McClements,"SpaceWire Higher Layer Protocols," Proceedings of International Astronautical Congress, Valencia, Sept 2006.
7. S. Parkes, "SpaceWire for Adaptive Systems", Proceedings of NASA/ESA Conference on Adaptive Hardware and Systems, 2008, Page (s) 77-82.

Missions and Applications 1

SPACEWIRE MISSIONS AND APPLICATIONS

Session: SpaceWire Missions and Applications

Short Paper

Doug Roberts

STAR-Dundee Ltd,c/o School of Computing, University of Dundee,

Dundee, Scotland, UK

Steve Parkes

University of Dundee, School of Computing, Dundee, DD1 4HN, Scotland, UK

E-mail: doug@star-dundee.com, sparkes@computing.dundee.ac.uk

ABSTRACT

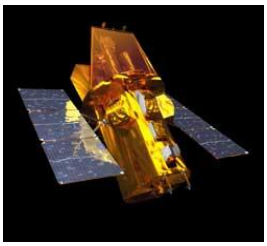
SpaceWire is an onboard data-handling network for spacecraft which connects together instruments, mass-memory, processors and telemetry sub-systems. It offers high-speed, low power, simplicity, low cost, and architectural flexibility. The growing heritage, available technology and capability of SpaceWire make it an ideal data-handling network for many future missions.

Since the SpaceWire standard was published in January 2003, it has been adopted by ESA, NASA, JAXA and RosCosmos for many missions and is being widely used for commercial and other spacecraft. High-profile missions using SpaceWire include: Bepi-Colombo, James Webb Space Telescope, ExoMars, Gaia, Astro-H, GOES-R, Lunar Reconnaissance Orbiter, Swift, PnPSat, and TacSat.

This paper provides an overview of some of the missions using SpaceWire. A summary of each mission is presented, its current status indicated, and the way in which SpaceWire is being used is described.

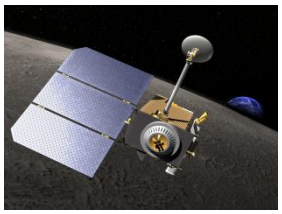
1. The Missions

1.1 Swift



NASA's Swift satellite is a multi-wavelength observatory which has been observing gamma-ray bursts since 2004 [1]. Planned to stay in low-Earth orbit for seven years, Swift has already exceeded its goal of observing 200 bursts, already having recorded over 400. SpaceWire is used to transfer information between the detectors and instrument readout electronics, and the instrument command and data handling unit.

1.2 Lunar Reconnaissance Orbiter

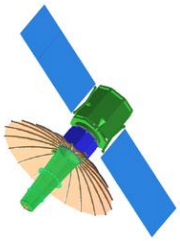


NASA's Lunar Reconnaissance Orbiter (LRO) was launched with the L-CROSS mission and has provided extensive high resolution data about the moon [2]. LRO is looking for resources and potential landing sites, and characterising the radiation environment. It also acted as a data relay for L-CROSS. Launched in late 2009, LRO is to remain in low polar orbit of the Moon for a year. LRO uses SpaceWire to connect its cameras, instruments, and communications, to the Command and Data Handling Computer.

1.3 Lunar Crater Observation and Sensing Satellite

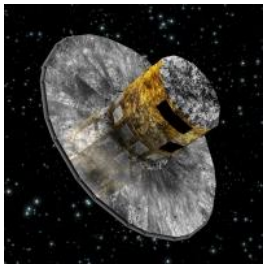
The Lunar Crater Observation (L-CROSS) was launched together with the Lunar Reconnaissance Orbiter in June 2009 [3]. L-CROSS completed its mission four months later, successfully discovering the existence of water at the Moon's South Pole. L-CROSS used SpaceWire in a very similar way to LRO [4].

1.4 TacSat-4



TacSat-4 is the fourth edition of the U.S. Military's tactical satellite series [5]. As with others in the series, TacSat-4 has real-time data requirements to provide military commanders with data. SpaceWire has been used as part of the payload-bus interface. It has been completed and is due to launch in August 2010.

1.5 ***Gaia***

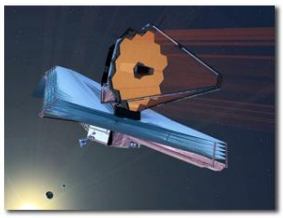


Gaia is a star mapping mission which aims to measure around a billion stars in and beyond our Galaxy, focusing on their positions, motions, and other physical characteristics [6]. Gaia is currently in the implementation phase and is due to launch in 2012, when it will travel to L2 and remain there until 2020. Gaia is using SpaceWire to transfer data from the optical terminal to the video processing units, the payload data handling unit, and the central distribution and monitoring unit.

1.6 ASTRO-H

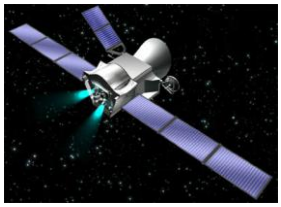
ASTRO-H is an X-ray observation satellite designed by JAXA [7]. It has six planned instruments, hard and soft X-ray telescopes and imagers, a soft X-ray spectrometer, and a soft gamma-ray detector. ASTRO-H is currently in the integration and test phases, and is due to launch towards the end of 2013, after which it is planned to stay in nominal operation into 2016. SpaceWire is being used as the primary data-handling system on the mission.

1.7 The James Webb Space Telescope



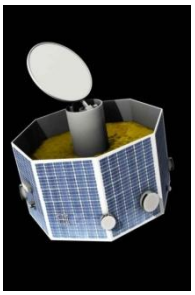
The James Webb Space Telescope (JWST) is an infrared observatory which is planned for launch in 2014 with an anticipated life of five years [8]. This ambitious NASA/ESA/CSA joint project aims to study deep into the past of our Universe by searching for distant traces of light. SpaceWire is being used extensively within the JWST, acting as the main data-handling bus.

1.8 BepiColombo Mercury Planetary Orbiter



ESA's Mercury Planetary Orbiter (MPO) will examine the surface and internal structure of Mercury [9]. MPO will be launched in 2014 together with the JAXA MMO as part of the combined BepiColombo mission. After six years it will arrive at Mercury to begin its year long nominal mission, with the option of a one year extension. SpaceWire is being used to transfer data from the payloads to the data processing unit.

1.9 BepiColombo Mercury Magnetospheric Orbiter



JAXA's Mercury Magnetospheric Orbiter (MMO) will look closely at Mercury's Magnetosphere [10]. MMO will separate from the BepiColombo mission when it reaches Mercury. MMO uses SpaceWire to connect the instruments to both data processing units and then the spacecraft system.

1.10 Magnetospheric Multiscale Mission

The Magnetospheric Multiscale mission (MMS) aims to study the microphysics of magnetic reconnection, energy particle acceleration, and turbulence [11]. NASA's mission, which orbits the Earth for around two and a half years, is due to launch late in 2014. SpaceWire is used as the primary data interface between the Payload and the Spacecraft.

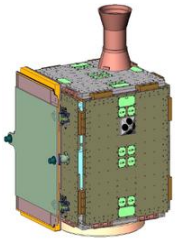
1.11 GOES-R

GOES-R is a geostationary environmental monitoring satellite which will replace the GOES-N, -O and -P series of satellites [12]. This joint NASA and NOAA mission contains six different instruments and will continue to provide the same service as previous GOES satellites. Due for launch in 2015, it will begin service in 2016 as GOES 14 retires, and has been designed to last to 2025. SpaceWire is used to transmit all sensor, telemetry, ancillary, command data, and time codes between the instruments and the spacecraft [13].

1.12 ExoMars

The joint ExoMars mission between ESA and NASA will provide further biological and environmental information about Mars, paving the way for robotic missions and human exploration [14]. The orbiter and the descent landing demonstrator are due to be launched in 2016, with the two rovers following in 2018. The rovers will use SpaceWire to connect the various cameras to the data-handling and processing system.

1.13 PnPSat-1



U.S.A.'s Air Force Research Laboratory's Plug-and-Play Satellite was designed to be the first satellite to demonstrate the ability of plug-and-play assembly of spacecraft [15]. It can reduce the integration time of satellites by using open standards and a simplified assembly process. PnPSat-1 features a SpaceWire Router to provide high performance interconnectivity between twelve endpoint ports, to which various equipment can be connected. It is currently awaiting a launch date.

2 References

[1] NASA, "Official NASA Swift Homepage", http://swift.gsfc.nasa.gov/docs/swift/swiftsc.html

[2] NASA, "Lunar Reconnaissance Orbiter", http://lunar.gsfc.nasa.gov/

[3] NASA, "LCROSS", http://lcross.arc.nasa.gov/

[4] NASA, "Glenn Rakow, SpaceWire Development Lead, Goddard Space Flight Center", http://greentechbriefs.com/component/content/article/3554?start=3

[5] Naval Research Laboratory via ScienceDaily , "TacSat-4 spacecraft complete and awaiting launch", http://www.sciencedaily.com /releases/2009/12/091201152646.htm

[6] ESA, "ESA – Space Science – Gaia Overview", http://www.esa.int/esaSC/120377 _index_ 0_m.html

[7] ISAS, JAXA, "ASTRO-H", http://astro-h.isas.jaxa.jp

[8] NASA, "The James Webb Space Telescope", http://www.jwst.nasa.gov/

[9] ESA, "ESA Science & Technology: BepiColombo", http://sci.esa.int/science-e/www/area/index.cfm?fareaid=30

[10] ISAS, JAXA, "Mercury Project: BepiColombo" http://www.stp.isas.jaxa.jp/mercury/

[11] Soutwest Research Institute, "MMS-SMART Home Page", http://mms.space.swri.edu/

[12] NOAA, NASA, DoC U.S.A., "GOES-R Home", http://www.goes-r.gov/

[13] W. H. Anderson, "The Geostationary Operational Satellite R Series (GOES-R) SpaceWire Implementation", International SpaceWire Conference 2007, Dundee, 17–19 September 2007

[14] ESA, "ESA – Aurora Programme – ExoMars", http://www.esa.int/SPECIALS/Aurora/SEM1NVZKQAD_0.html

[15] D. Fronterhouse, J. Lyke, "Plug-and-Play Sattellite (PnPSat): Demonstrating the Vision", http://spacewire.computing.dundee.ac.uk/proceedings/Presentations/Missions%20and%20Applications%201/fronterhouse.pdf

THE GEOSTATIONARY OPERATIONAL SATELLITE R SERIES SPACEWIRE BASED DATA SYSTEM ARCHITECTURE

Session: SpaceWire Missions and Applications
Long Paper

Alexander Krimchansky - NASA Goddard Space Flight Center
William H. Anderson, Craig Bearer - MEI Technologies
E-mail: Alexander.Krimchansky@nasa.gov, William.H.Anderson@nasa.gov, Craig.M.Bearer@nasa.gov

ABSTRACT

The GOES-R program selected SpaceWire as the best solution to satisfy the desire for simple and flexible instrument to spacecraft command and telemetry communications. Data generated by GOES-R instruments is critical for meteorological forecasting, public safety, space weather, and other key applications. In addition, GOES-R instrument data is provided to ground stations on a 24/7 basis. GOES-R requires data errors be detected and corrected from origin to final destination. This paper describes GOES-R developed strategy to satisfy this requirement.

1. INTRODUCTION

The Geostationary Operational Environmental Satellite-R Series (GOES-R) program is a key element of the National Oceanic and Atmospheric Administration's (NOAA) operations. As such, the GOES-R and follow on series of satellites will be comprised of improved spacecraft and instrument technologies, which will result in more timely and accurate weather forecasts, and improve support for the detection and observations of meteorological phenomena that directly affect public safety, protection of property, and ultimately, economic health and development. The first launch of the GOES-R series satellite is scheduled for 2015. The GOES-R spacecraft uses European Cooperation for Space Standardization (ECSS) SpaceWire [1] for the transfer of sensor, telemetry, ancillary, command, time code, and time synchronization information between instruments and the spacecraft.

2. RELIABLE DATA DELIVERY PROTOCOL

GOES-R project has developed a Reliable Data Delivery Protocol (GRDDP) that is based on SpaceWire capabilities for link connection and re-connection, error detection, virtual channels and routing. This protocol has been presented to and accepted by the SpaceWire Working Group [2] and assigned a Protocol ID (PID) 238. GRDDP also known as PID 238 does not attempt to duplicate or improve on the considerable capabilities provided by SpaceWire. This protocol builds on top of SpaceWire the ability to recover lost packets, reorder packets, and to ensure to higher level processes that packets are as error free as possible.

GOES-R requirements for PID 238 are to utilize SpaceWire capabilities to provide a packet delivery protocol that is able to detect and recover lost packets. The protocol is also required to be flexible so that it can be adapted as needed to different host data throughput requirements and resources. PID 238 intentionally does not specify an implementation. It

defines a set of capabilities, but does not require that all capabilities be implemented for all applications.

PID 238 is based on the concept of "Virtual Channels" similar to the virtual channels identified in the SpaceWire specification. Any number of virtual channels can coexist on a single SpaceWire link. In PID 238, all channels are completely independent. PID 238 defines virtual channels as a pair of Transport End Points (TEPs). Each TEP is identified by a SpaceWire Logical Address (SLA) and interfaces to a SpaceWire link to send and receive PID 238 packets. Each PID 238 virtual channel transmits data packets in only one direction, thus each channel consists of one Transmit TEP and one Receive TEP. PID 238 protocol is completely specified in terms of the behavior of a Transmit TEP and a Receive TEP. A host will have only one TEP for each virtual channel that it supports. Note that a Transmit TEP sends data packets, but also receives acknowledge packets from the remote Receive TEP. A Receive TEP receives data packets, but also transmits acknowledge packets.

PID 238 specifies a packet format that is consistent with the standard SpaceWire packet. Specifically a packet terminated with an End of Packet (EOP) or Error End of Packet (EEP) character. However, where the SpaceWire standard allows a path address of zero or more bytes, PID 238 requires that exactly one destination SLA be delivered to PID 238 logic at the destination. That destination SLA identifies the virtual channel that is to receive the PID 238 packet. PID 238 does not dictate packet routing through a SpaceWire network be it point-to-point or composed of multiple routers. Packet routing is handled by the SpaceWire layer. PID 238 does not try to improve on SpaceWire packet routing. The one-byte destination SLA is sufficient to get a SpaceWire packet to its destination through a variety of network configurations.

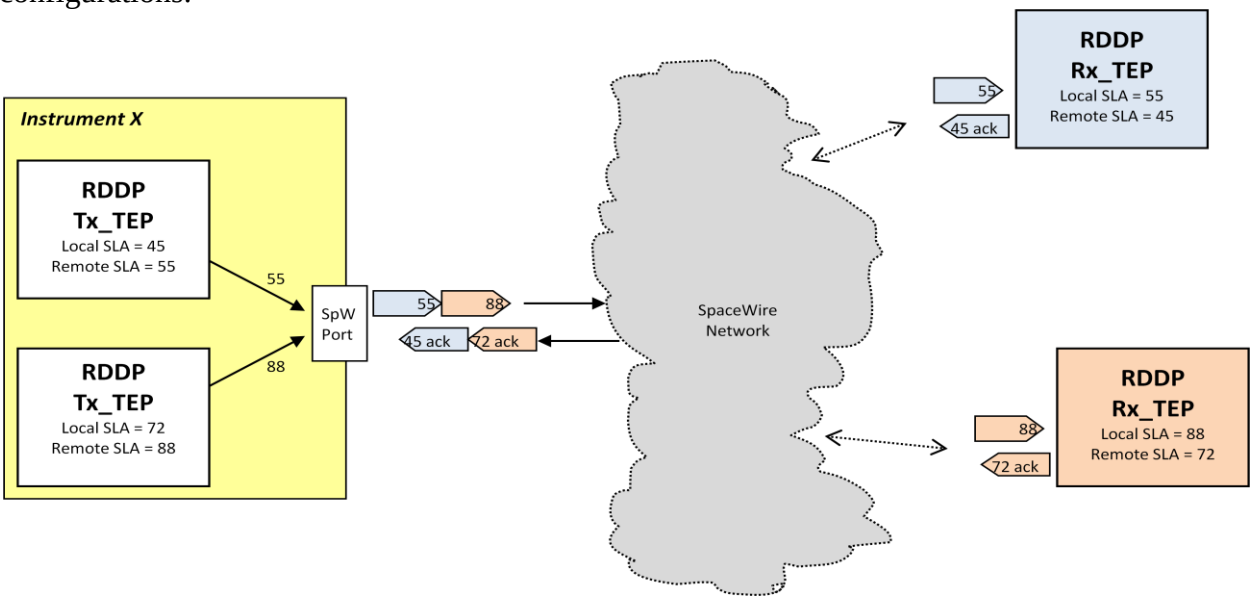


Figure 1. PID 238 Packet Routing

Figure 1 illustrates PID 238 perspective for packet routing. Each of the several virtual channels that may reside on a host has associated with it the destination SLA for that channel's remote TEP. For PID 238, there is no "path" to the remote TEP, only a destination SLA. PID 238 does not have, and does not require, knowledge of how a packet gets to its destination. This keeps PID 238 simple, and allows any implementation using this protocol

to remain independent of possible SpaceWire network changes, and unaffected if a remote TEP for a channel is re-configured to reside on a different host. Figure 2 shows PID 238 format and details can be found in the protocol's document [3].

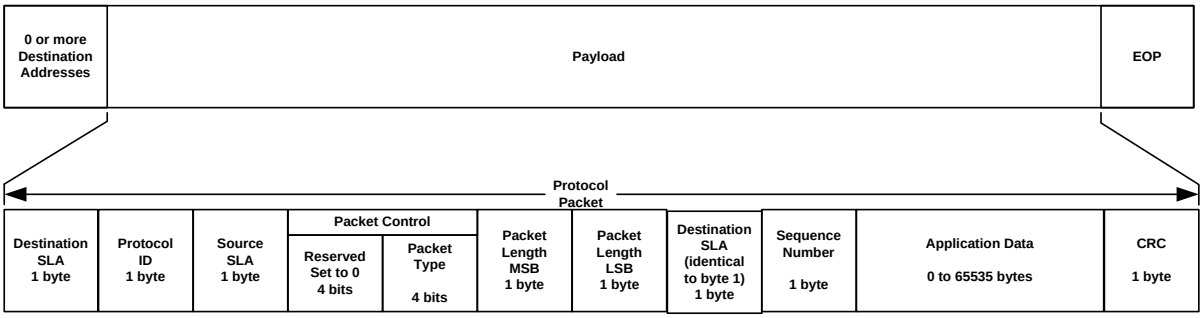


Figure 2. PID 238 Packet Format Inside a SpaceWire Packet

Key features of PID 238 are:

- Defines 4 kinds of packets
- Data Packets have Sequence Numbers from 0 to 255
- ACK Sequence Numbers must match the source Data Packet
- The Urgent Message Sequence Number is always 0
- The Reset Packet Sequence Number is always 0

3. PID 238 OPERATION OVER SPACEWIRE

The TEP for each channel is in one of three possible states Closed, Enabled, or Open. By default, all channels are in the Closed state until the host sets the channel to Enabled. A channel that is in the Closed state will not transmit any packets, and will not acknowledge or process any packets received. When in the Enabled state, a transmit TEP will send only Reset packets to the remote receive TEP. When an Enabled receive TEP gets the Reset Packet, it changes to the Open state and then sends an ACK for the reset. When the transmit TEP receives the ACK it changes to the Open state. The Reset and ACK packets are what lets both ends of the channel know that the other end is "open", or active. The reset also serves to synchronize the "next" Sequence Number expected for both ends of the channel. Note that only the transmit side can open a channel. An enabled receive TEP can only sit back and wait for the transmitter to become active (or enabled) and send the Reset Packet. Until the transmitter is ready to start sending Data Packets there is no reason for the receiver to do anything.

PID 238 provides several capabilities that work together to provide the reliable data delivery that is required for GOES-R.

PID 238:

- Detects lost packets by using positive acknowledge for each packet transmitted.
- Recovers lost packets using an ACK timeout and retransmit.
- Re-orders received packets and removes duplicates using sequence numbers.
- Maximizes data packet throughput using a sliding window range of sequence numbers that allows a transmitter to continue sending packets while waiting for acknowledgements

Each of these capabilities can be independently adapted for each channel so that PID 238 can be adapted to the data throughput and reliability requirements for each data stream.

PID 238 uses a positive acknowledge for each packet transmitted. If an ACK is not received within a timeout interval, that packet is retransmitted. After a maximum number of retries have been exhausted for a packet, the transmitter will declare that virtual channel (and only that channel) closed. The requirement for the receiver to acknowledge each packet allows the transmitter to detect lost packets if an ACK is not received. On the receiver end, a packet is acknowledged if it has a valid PID 238 header and Cyclic Redundancy Check (CRC) character.

In the case where data packets represent a "current" value that changes at a high rate. In this case there is no point in re-transmitting an un-acknowledged old packet when a new packet is available. PID 238 defines an "Urgent Message" packet type that does not need to be acknowledged, and is delivered to the host at a higher priority than the normal data packets. In order to keep things simple, the Urgent Messages do not require a separate channel. Urgent Messages can be intermixed with normal data packets on a single channel.

In order to maximize throughput, PID 238 defines a moving window range of sequence numbers that allows the transmitter to transmit ahead while waiting for ACKs. The size of the window range limits the number of packets that can be transmitted by a Transmit TEP while waiting for an ACK. At the Receive TEP, the window is used to eliminate duplicate packets, and to ensure packets are delivered in correct sequence to the host.

A channel that is required to transmit high rate data would use a large window so that a large number of data packets can be sent ahead of receipt of the ACKs. If an ACK is not received within a timeout interval, the transmit TEP will retransmit only that packet. The Receive TEP implements the same size window to order the packets delivered to the host and will deliver the retransmitted packet in correct sequence. The moving window, and the associated memory buffers required for retransmission of old packets and for re-ordering received packets does require some additional logic and memory resources. The amount of memory required to buffer moving window data packets depends on the maximum packet size and the window size.

In a case where a PID 238 channel does not require a high data rate, the window size can be set to one. A window size of one provides a synchronous transfer where the transmitter waits for each packet to be acknowledged before sending another. Finally the simplest PID 238 implementation is used when data is updating at some known rate and fresh data will quickly replace lost or stale data. This is the Urgent Message service. Urgent Message does not require moving window processing or an ACK packet. It is straight forward fire-and-forget.

PID 238 Mode	Usage
Full Mode Window Size >1	High Data Rate
Full Mode Window Size = 1	Low Data Rate
Urgent Message Window Size = 0	Low Latency

Table 1. PID 238 Usage

In summary, PID 238 specifies the behavior of Transmit and Receive TEPs. Transmit TEPs will transmit only Data and Reset packets. A transmit TEP will never receive a data packet or a reset, and will never transmit an ACK. A Receive TEP will transmit only ACKs, and will receive only data packets or resets. A Receive TEP will never transmit a Data or Reset packet. The rules for when a Transmit TEP sends Data or Reset packets and a Receive TEP sends an ACK are not complicated. Much of PID 238 flexibility for adapting to different instrument requirements has to do with the number of transmit or receive channels, whether Urgent Messages are used, and the window size selected. An instrument with very low rate data requirements might elect to implement one transmit TEP and one receive TEP (only one channel in each direction), and to use a window size of one for both the transmit and receive channels. In this case, PID 238 would provide synchronous data transfers where each packet transmitted must be acknowledged before another can be sent. With a window size of one, the need for transmit or receive buffers is minimal, and there is no need to implement the logic to re-order packets. At a minimum, PID 238 provides multiple virtual channels that can be independently routed via the SpaceWire network, and improved error detection through the CRC.

4. SPACEWIRE AND PID 238 TEST SYSTEM

GOES-R project has maintained a SpaceWire test lab [4], initially to validate PID 238, to evaluate varying the parameters that might affect performance, and for validating different proposed options and configurations that could be used on the GOES-R spacecraft. The GOES-R test system has 2 SpaceWire implementations. The first is a GOES-R test card utilizing the British Aerospace (BAE) SpaceWire Application Specific Integrated Circuit (ASIC). And the other is a Xilinx Field Programmable Gate Array (FPGA) commercial off-the-shelf card. The FPGA card is programmed with a modified version of the Goddard SpaceWire core that removed the SpaceWire worm hole router. Additionally, the FPGA core provides additional diagnostic and error injection capabilities. These test cards reside in off-the-shelf Windows workstations and the initial test system is shown in Figure 3.

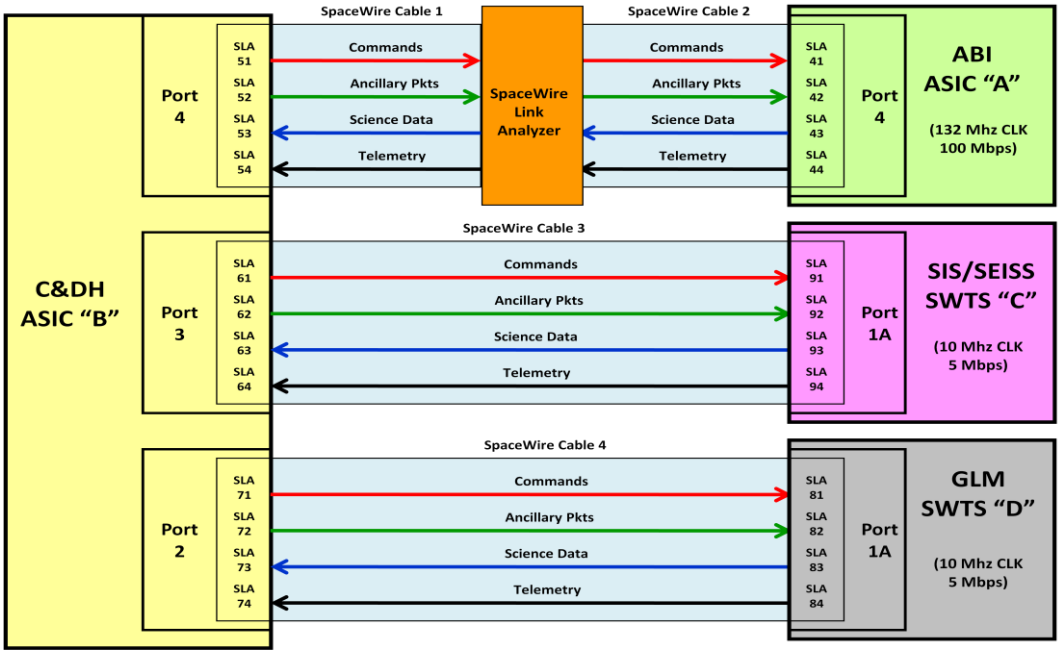


Figure 3. PID 238 Over SpaceWire Original Test System

In all cases, where the BAE SpaceWire ASIC is used to simulate an instrument or the Command and Data Handling (C&DH) system, the same Embedded Microcontroller (EMC) code, including tunable parameters, is used for each test article. PID 238 software implementation is approximately 600 lines of code. Depending on the packet rate, Central Processing Unit (CPU) utilization is in the single digit percentage range on the EMC clocked at 33Mhz. The EMC is capable of operating at clock rates up to 80Mhz.

The probability of an error occurring during transmission of a packet depends on the size of the packet. The ability to recover a lost packet by retransmitting also depends on the size of the packet. Small packets require retransmission of only a small packet making it easier to insert into a data stream. Where large packets have a larger impact and may require more bandwidth margin. GOES-R testing has successfully recovered lost packets, of various sizes, when the normal data stream uses over 90% of the available bandwidth.

5. Next Steps

Of the 5 GOES-R instruments, 2 have implemented PID 238 in FPGAs, and the other three have implemented the protocol in software on the embedded microcontroller in the BAE SpaceWire ASIC. Each of the GOES-R instruments are implementing the SpaceWire and PID 238 interface as a point-to-point architecture. Modeling the proposed spacecraft data system has shown no changes are required in any instrument implementation including the addition of several SpaceWire routers.

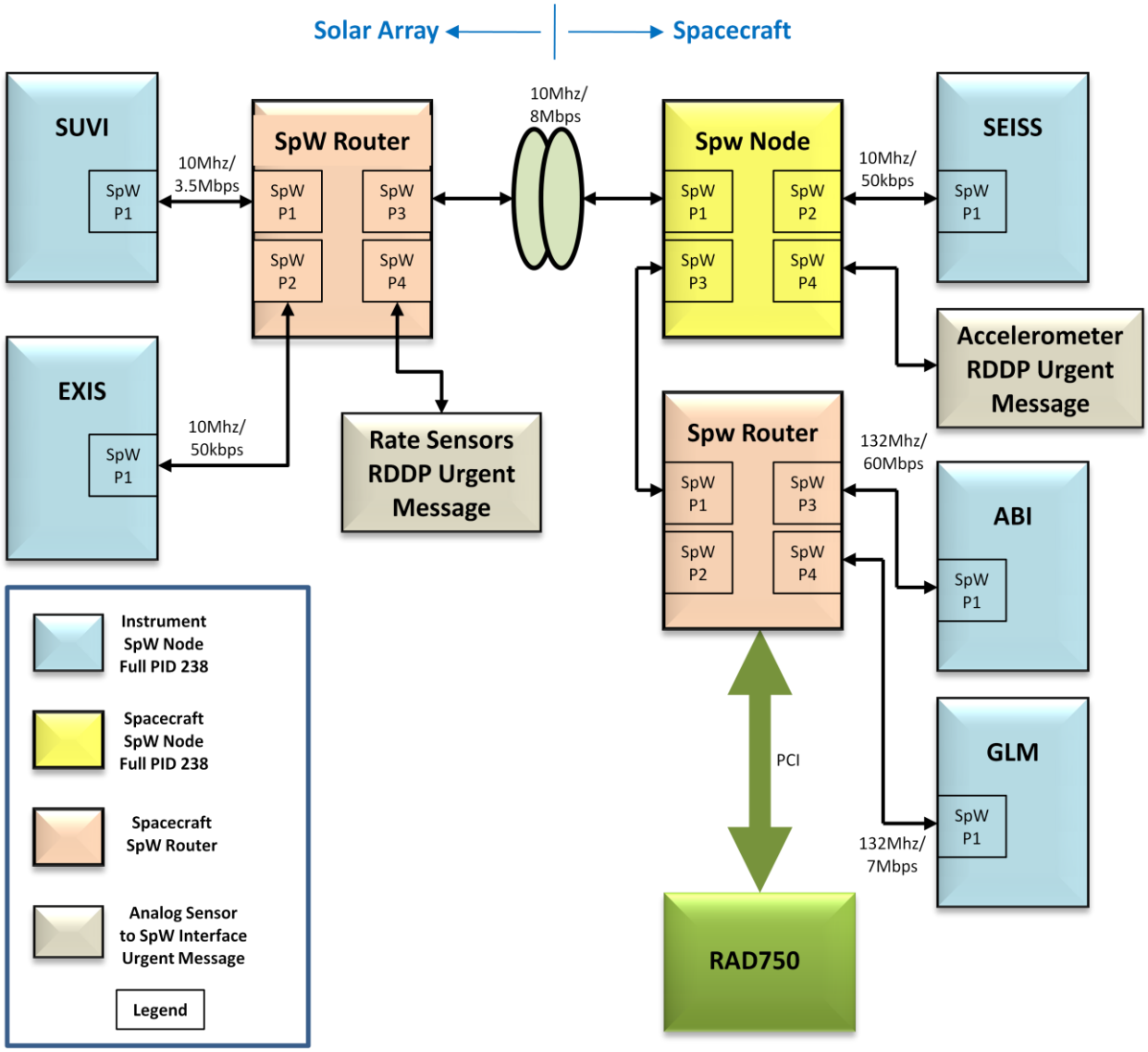


Figure 4. Simplified Spacecraft Design with Multiple SpaceWire Routers

The most simple instrument with very small data throughput requirements and minimal processor resources, the largest instrument with the highest data throughput requirements, and the spacecraft C&DH that interfaces to them all have implemented PID 238 to the same specification. All of the instruments as well as the spacecraft recognize a common method for detecting and recovering data link errors and lost packets. GOES-R has several years experience exercising SpaceWire and PID 238 in a variety of configurations. In all cases SpaceWire and PID 238 have been found to be robust, efficient, and flexible. PID 238 over SpaceWire is documented, tested, and available for use in space data system applications.

6. CONCLUSION

PID 238 was developed to satisfy data system requirements for all GOES-R instruments. The instruments have a wide range of electronics implementations from simple to complex, and a wide range of data rates. SpaceWire transmit clock rates operate at either 10MHz or 132MHz. GOES-R instrument data rates ranging from 50kb to 66MHz are easily managed by the combination of PID 238 over SpaceWire. Many parameters of PID 238 can be tuned to match the reliability requirements and a node's ability to support the required complexity. PID 238 has proven able to adapt to those capabilities and data rates due to its inherent flexibility. PID 238 is documented and extensively tested. It is available and ready to be applied to SpaceWire applications.

7. REFERENCES

1. European Cooperation for Space Standardization, ECSS-E-50-12A, "SpaceWire – Links, Nodes, Routers and Networks", 24 January 2003

2. European Cooperation for Space Standardization, ECSS-E-ST-50-51C, "SpaceWire Protocol Identification", 5 February 2010

3. NASA Goddard Space Flight Center GOES-R Project "GOES-R Reliable Data Delivery Protocol", 417-R-RTP-0050 Version 2.1, 16 January 2008

4. William Anderson MEI Technologies Inc., GOES-R Project, "Reliable Data Delivery Protocol (RDDP)", 2006 MAPLD International Conference, Washington, D.C., September 25, 2006

SPACEWIRE DRIVEN ARCHITECTURE FOR THE ASTRO-H SATELLITE

Session: SpaceWire Missions and Applications

Long Paper

Masanobu Ozaki, Tadayuki Takahashi, Motohide Kokubun, Takeshi Takashima, Hirokazu Odaka

Institute of Space and Astronautical Science/JAXA, 3-1-1 Yoshinodai, Chuou-ku, Sagamihara, Kanagawa 252-5210, Japan

Masaharu Nomachi

Osaka University, 1-1 Machikaneyama, Toyonaka, Osaka 560-0043, Japan

Takayuki Yuasa

The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-0033, Japan

Iwao Fujishiro

SHIMAFUJI Electric Incorporate, KC building 5F, 8-1-15 Nishi-Kamata, Ota-ku, Tokyo 144-0051, Japan

Takayuki Tohma

NEC Corporation, 1-10 Nisshin-cho, Fuchu, Tokyo 183-8501, Japan

Hiroki Hihara

NEC TOSHIBA Space Systems, Ltd., 1-10 Nisshin-cho, Fuchu, Tokyo 183-8501, Japan

Kazunori Masukawa

Mitsubishi Heavy Industries, Ltd. Nagoya Guidance & Propulsion systems works, 1200 Higashi-Tanaka, Komaki, Aichi 485-8561, Japan

E-mail: ozaki@astro.isas.jaxa.jp

ABSTRACT

ASTRO-H is a very large scale and complicated spacecraft as a scientific mission. In order to realize the project with limited resources, SpaceWire is used as the information network base, and many elements such as the network architecture, standard nodes and ground support equipments have been designed. The design is adaptive for not only ASTRO-H but other missions: in JAXA, the small satellites project will use the common design architecture.

1 INTRODUCTION

ASTRO-H[1] is the 6th Japanese X-ray astronomy satellite, which is scheduled to be launched in 2014. The requirements for the satellite controlling units, such as of

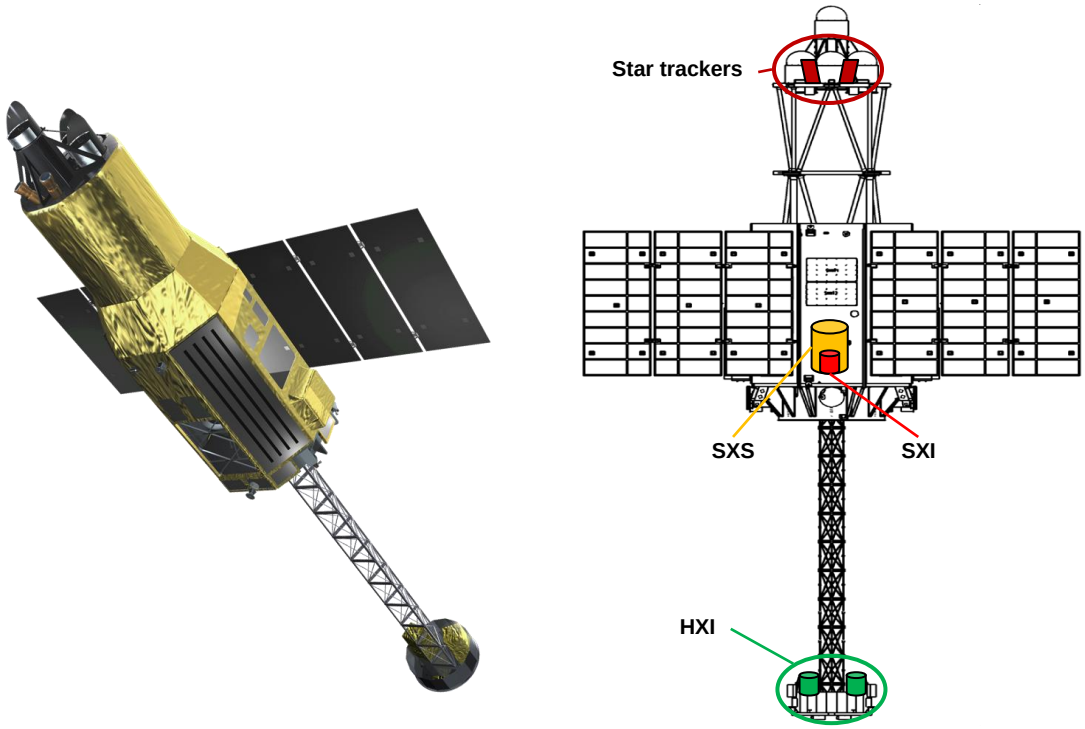


Figure 1: A CG image (left) and a schematic frame drawing (right) of the ASTRO-H satellite. The SpW network covers not only the main body but also the extendable optical bench (EOB) plate and the fixed optical bench (FOB) top plate.

system management, telemetry-and-command handling and attitude controlling, are more complicated than for past scientific satellites. In addition to this, the satellite carries 4 different kinds of scientific payloads, X-ray micro calorimeter (SXS), X-ray CCD camera (SXI), hard X-ray imaging spectrometer (HXI) and soft gamma-ray detector (SGD), each of which has different type of sensor and on-board data processing scheme from others. They make the satellite structure complex and force multiple companies to be deeply involved in the interface coordination, which usually introduces long negotiation, development and integration phases and lead the project to a cost-consuming way.

SpW[2] motivated us to define new system architecture to solve these problems, which is applicable not only to ASTRO-H but also to other projects. We thus build up a new standard that defines network protocol, router, the standard computer architecture and the standard I/O module, which are constructed on the SpW standards and can be implemented by multiple companies. In addition to them, we also defined a handling scheme of the CCSDS Space Packet on the network. In parallel to the standard definition process, we organized a SpW user community with other JAXA satellite projects, which works as the technical forum among the standard developers, component developers and users.

2 On-board architecture

The ASTRO-H information-exchange framework is wholly SpW base. This consists of the network, the data format and each component (i.e., SpW node).

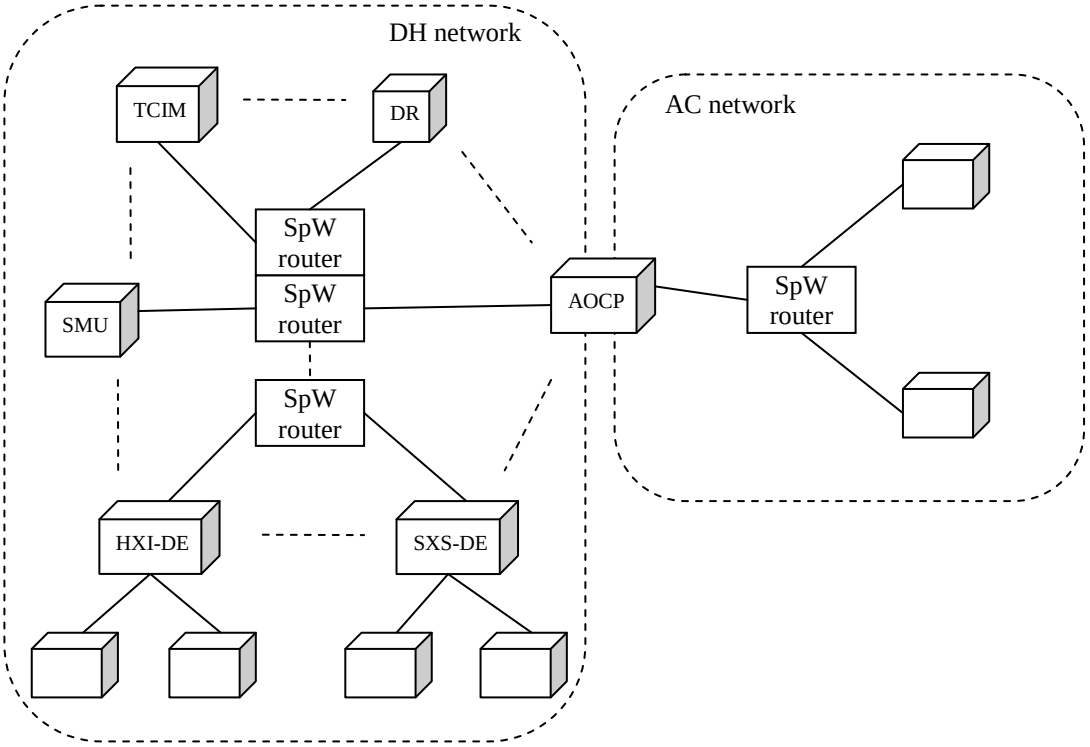


Figure 2: A schematic diagram of the physical network topology. There are two independent star topology networks.

2.1 REQUIREMENTS

The ASTRO-H observatory is expected to generate about 12Gb of data par day, which corresponds to about 140 kbps, which is not very large value for a modern data transfer line. This is, however, a result of on-board data reduction process, and the original data generation rate is 10-100 times larger than this. In order to handle the original data without any loss, the network must have a well-designed topology.

2.2 NETWORK

The SpW network covers whole the satellite, whose length is about 14 m on orbit and the core part shapes octagonal column inside which most of the components are attached. On the base panel inside the column, SXS, SXI and the fixed optical bench that supports the X-ray mirrors, the star trackers and the optical alignment system are settled. The extendable optical bench is attached at the backside of the base panel and supports HXIs. SGDs are placed at the outside of two panels. Figure 1 shows a schematic drawing of the satellite structure.

2.2.1 TOPOLOGY AND DIRECTION

The ASTRO-H SpW network consists of two physically-independent subnets: one is the DH (data handling) network that is controlled by the satellite management unit (SMU) and includes data recorders (DRs) and telecommunication components, and the other is the AC (attitude control) network that is controlled by the attitude and orbit control processor (AOCP). AOCP is also connected to the DH network, but no SpW packet, even TI packet, is forwarded between two networks. All the AC network components other than AOCP are sensors or actuators, and all the traffic in the

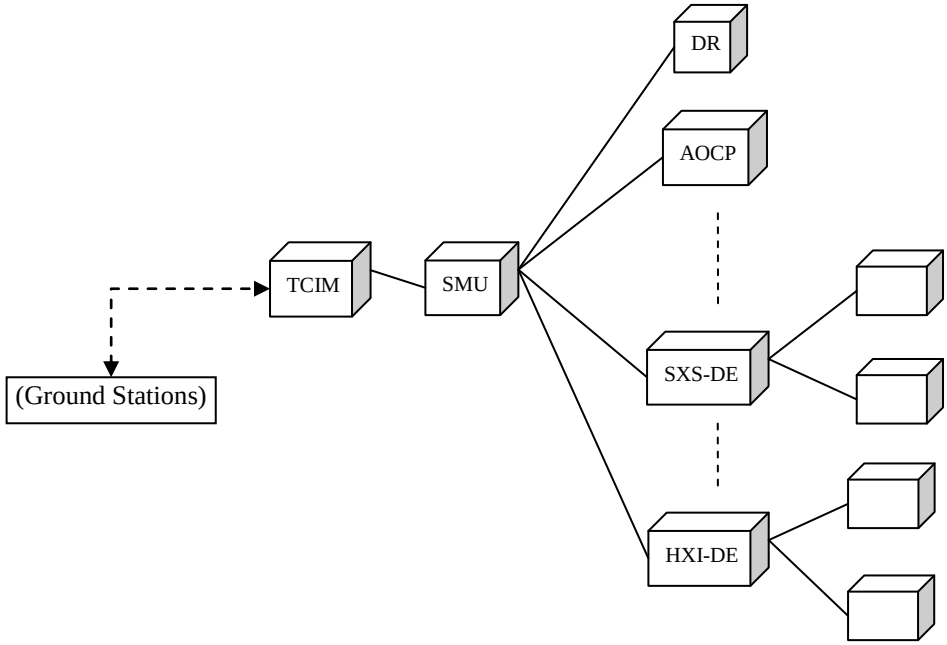


Figure 3: A schematic diagram of the logical topology of the DH network. The network has tree topology, and the communication is basically held between a parent and a child in normal operation.

network is initiated by AOCP. Figure 2 shows schematic diagram of physical network topology.

The components of each network are basically connected to the SpW routers that are based on the crossbar switch technology and form the physical networks of star topology: multiple SpW communication thus can be held simultaneously with no packet collision. All the SpW components are connected to two or more different SpW components or routers, and single failure of any connection can be substituted by an alternative path. The link speed from SpW routers and SMU is 25 MHz.

The logical structure of the ASTRO-H SpW network is, on the other hand, tree topology starting from the telemetry command interface module (TCIM), and all the SpW transaction are initiated from the root-side of the network. TCIM communicates with SMU, and SMU sends out all the commands to other components in normal operation. Most of the communication except for diagnostic or initialization ones are held between parent and child nodes and initiated from the parent: other kind of communication is physically possible, but forbidden except for ones between SMU and TCIM or DR and other components. The reason of the latter exception is that most of the telemetry data generated by payload instruments are not edited by SMU and can be transferred to the ground stations directly. Figure 3 shows schematic diagram of logical network topology.

All the traffic between SMU and other components in the DH network is thus initiated by SMU. The traffic is categorized into command and telemetry in the application layer: the former contents flow from SMU to other components, and it is natural that SMU initiate the transaction. The latter are, on the other hand, generated from other components in most cases. In order to transfer such contents, we chose the architecture that the destination side (SMU or DR) collects each component's output with periodic polling by SMU.

2.2.2 TIME CODE

The TI packet of the DH network is generated by SMU. SMU receives the reference clock and time information from a GPS receiver via an exclusive line and SpW, respectively. The TI packet frequency is 64 Hz.

2.3 PROTOCOLS

All the SpW communication inside ASTRO-H uses RMAP[3]. With this protocol, all the components can have network-transparent accessibility, which is quite useful for diagnostic and initialization not only in R&D phase but also even on orbit. The contents carried by the RMAP access is classified into two: Space Packets[4] and raw data.

2.3.1 CCSDS ON RMAP

Commands to all the components and telemetry from major instruments such as scientific payloads are transferred as Space Packets on RMAP. All the command packets are sent out from SMU, and the telemetry packets are generated by multiple components. The latter ones are sent out to SMU or DR, and transferred to the ground stations at the end.

Each command packet is transferred by independent RMAP transaction. Multiple telemetry packets, on the other hand, can be transferred by one RMAP transaction.

2.3.2 PIM ON SPW

In case of non-intelligent components that cannot generate Space Packets, the telemetry data are generated as raw binary data on the RMAP memory space. The memory map is based on that of the peripheral interface module (PIM) that has been used for previous Japanese scientific satellites[5]. The data are periodically read and assembled as Space Packets by SMU.

2.4 COMPONENTS

As described in Section 1, the main motivation of introducing SpW to the ASTRO-H project is to make the satellite system as simple as possible. We accordingly developed several kinds of SpW devices that can handle not only the physical layer but also up to RMAP level. The largest component is SpaceCube, which is a computer architecture that can handle RMAP/SpW and work as a stand alone machine. Another group consists of embedded devices that bridge between components' internal bus and RMAP memory space: they cannot work without other devices such as memories or sensors, but make old architecture payloads as SpW components.

In all cases, the signal ground of all the SpW devices in all the components are connected to the satellite body explicitly because the LVDS devices, which are used in the physical layer of SpW, generally have low tolerance to over voltage on the I/O pins and it is essential to align the signal ground levels among all the SpW components.

2.4.1 SPACECUBE

SpaceCube is a computer architecture for spacecraft information system, and has following features: (1) open architecture based on T-Engine, (2) SpW connection, (3)

source-code level compatibility in the application layer between different implementation and (4) small, low mass, low power consumption and low cost by introducing recent results from commercial products. The original SpaceCube, which is sometimes called as SpaceCube1, is developed by Shimafuji and JAXA. The ASTRO-H on-board computers connected to the SpW network are its derivatives or successors, which are implemented by NEC and MHI independently.

2.4.2 EMBEDDED RMAP DEVICES

For small components such as simple sensors or actuators, a CPU-based system such as SpaceCube is too complicated in most cases. In order to connect such components to the SpW network, we developed IPs that can be implemented in an FPGA and ASICs. They can handle not only SpW connection and network but also RMAP, and are embedded in many components, such as the power control unit and SpW I/O boards used for the scientific payloads. They are also implemented by multiple companies independently, as the SpaceCube computers do.

3 SUPPORTS FOR DEVELOPMENT

As SpW is on the cutting edge of the spacecraft information technology, we are taking both technical and social approach to make use of it. Both are depending to each other and difficult to be separated clearly, but we can pick up some prominent cases: the ground support equipments (GSE) supplied by the system integrator and the user community.

3.1 GSE

As the spacecraft information system is not only an aggregation of communication channels but a highly organized system managed by SMU, a GSE that act as the network manager is essential for the payload instrument development. For this purpose, the satellite integrator (NEC) and JAXA are preparing two kinds of SMU simulators. The minimum simulator is called “SMU Sim Light” and has abilities to send out Space Packet commands, collect raw binary data from target components and receive Space Packet telemetry data. The simulator consists of a commercial grade SpaceCube computer, the on-board software and a PC that works as the console.

The full configuration simulator is called “SMU Sim”. This system contains whole the ground control system equivalent to that used for the real spacecraft control, and can emulate not only SMU but also the control system used in the integration test phase. The component developers thus can carry out interface tests in the development phase independently before the system integration test.

3.2 COMMUNITY

In order to develop SpW based system, we formed a SpW user group in Japanese space community where the members share the basic knowledge such as the startup procedure of the SpW system development and a reference implementation of SpW and RMAP software on SpaceCube. The member is not limited to JAXA and aerospace companies, and the group works as a gateway to commercial base SpW instruments and ground-based applications.

The ASTRO-H SpW components are designed as adaptive for other missions. In ISAS/JAXA, the small satellites project is running in parallel to ASTRO-H, and many of the DH and AC network components will be used in that.

4 CONCLUSION

We are developing the ASTRO-H satellite with using the SpW technology as the main information framework, and many elements such as the network architecture and the standard components are designed. They will also used for other Japanese future scientific space missions.

5 REFERENCES

1. Tadayuki Takahashi, “The NeXT mission”, SPIE meeting 7011 "Space Telescopes and Instrumentation II: Ultraviolet to Gamma Ray 2008", Marseille (2008)

2. ECSS, “SpaceWire”, ECSS-E-50-12C

3. ECSS, “SpaceWire Protocols”, ECSS-E-50-11C

4. CCSDS, “Space Packet Protocol”, CCSDS 133.0-B-1

5. Hiroki Hihara, “Common communication scheme using common memory mapping over RMAP”, 13th SpaceWire working group meeting, ESTEC (2009)

OVERVIEW OF IMPLEMENTING SPACEWIRE IN OBSERVATION SATELLITES FROM THALES ALENIA SPACE

Session: SpaceWire Missions and Applications

Long Paper

Alain Girard, Antoine Provost-Grellier, Jean Nodet, Philippe Desmet, Patrice Cossard

Thales Alenia Space,

Etablissement de Cannes,

100 bd du Midi, 06156 Cannes La Bocca, FRANCE

E-mail: : alain-felix.girard@thalesaleniaspace.com, antoine.provost-grellier@thalesaleniaspace, jean-christian.nodet@thalesaleniaspace.com, philippe.desmet@thalesaleniaspace.com, patrice.cossard@thalesaleniaspace.com

ABSTRACT

Thales Alenia Space is implementing SpaceWire technology in most future space projects for observation missions. Today, three kinds of space applications are currently based on SpaceWire architectures for payload data handling, with low earth orbit observation satellites, planetary exploration carriers and geostationary observation satellites.

All various missions are briefly described and compared for SpaceWire implementation, showing how the SpaceWire use and perimeter is increasing from interfaces standardization up to interfaces optimization by merging of mission and configuration command/control data for exceeding limits of today architectures :

- the Sentinel-3 satellites for the provision of operational marine and land services, based on optical and microwave Earth observation payload,
- the ExoMars mission for Entry, Descent and Landing Module (EDM) of a payload on the surface of Mars,
- the MTG system will provide Europe's National Meteorological Services for both meteorological and climate applications.

1 INTRODUCTION

Thales Alenia Space is implementing SpaceWire technology in most future space observation missions. Today, three kinds of space applications are based on SpaceWire architectures for payload data handling, with low earth orbit observation satellites, planetary exploration carriers and geostationary observation satellites.

Main missions are briefly described and compared for SpaceWire implementation, listing the advantage and the criticality in implementing SpaceWire. The comparison on how the SpaceWire is used, shows that the SpaceWire perimeter is increasing from

interfaces standardization, through interfaces optimization by merging of mission and configuration command/control data up to allowing to exceed limits of today architectures.

Since SpaceWire is now mature and suitable for space high speed communication, it is natural for Thales Alenia Space to implement it now in its first European LEO observation mission, i.e. GMES Sentinel-3. As a first SpW application at system level for Thales Alenia Space, the use of SpW links focus on high speed mission data distribution for which SpW bring advantages compared to previous solutions.

Then thanks to the preliminary achievements with LEO satellite, the use of SpW might be extended for planetary exploration orbiter. This solution is presently under analysis.

Finally the use of SpW for GEO observation satellite is obvious as for LEO ones, since such missions embed high speed instruments. Today, these missions are dedicated to operational meteorology. Furthermore, GEO meteorology requires continuous handling of high speed mission data flows without allowed outage for various payload modes based on multiple instrumen's, for which routing capability and fast imaging configuration command/control is required : SpW technology is today offering these services thanks to the routing and full-duplex capabilities.

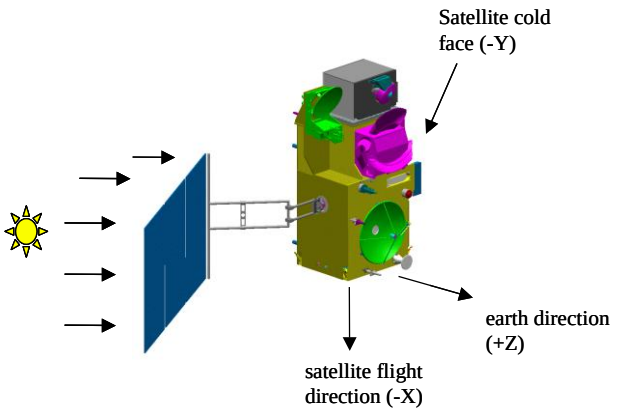
2 LEO OBSERVATION MISSIONS WITH SPACEWIRE

Thales Alenia Space is the prime contractor of the GMES Sentinel-1 and 3 programs. The four Sentinel-1 and 3 satellites embed several instruments which generate high rate data stream and since their development calendars fit with SpW technology maturity, it was the good targets for Thales Alenia Space to start SpW implementation at system level.

As first SpW implementation on a complete satellite's payload and since SpW is first suitable for high speed communication, Thales Alenia Space efforts focus on high speed mission data distribution from instruments to the mass memory for RF downlink.

2.1 SENTINEL-3 MISSION

In the frame of the Global Monitoring for Environment and Security program (GMES), the Sentinel-3 is a European polar orbit satellite system for the provision of operational marine and land services, based on optical and microwave Earth observation payload.



The Sentinel-3 Data Handling

architecture design has been driven by a) minimized development risks, b) system at minimum cost, c) operational system over 20 years.

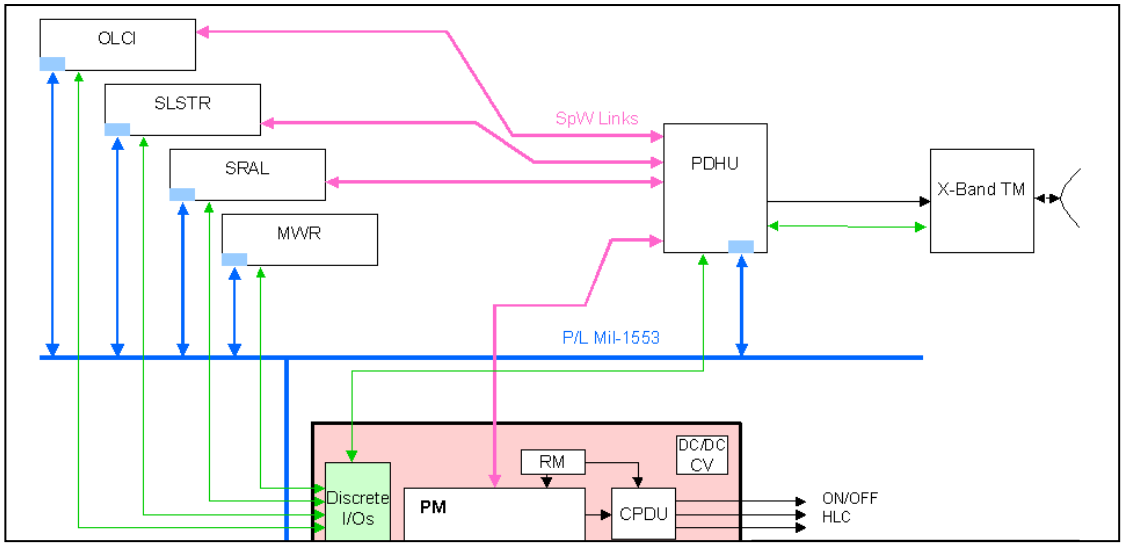
This has led to design architecture as robust as possible using a single Satellite Management Unit SMU computer as the platform controller, a single Payload Data-Handling Unit for mission data management, and to reuse existing qualified heritage.

The payload accommodates 6 instruments, sources of mission data. The 3 high rate instruments provide mission data directly collected through the SpW network, while the low rate instruments are acquired by the central computer for distribution through the SpW network to the mass memory. The PDHU acquires and stores all mission data for latter multiplexing, formatting, encryption and encoding for download to the ground.

SENTINEL 3 Satellite main features	
Instruments	Sea and Land Surface Temperature sensor (SLSTR) Ocean and Land Colour sensor (OLCI) Altimeter (SRAL) Microwave radiometer (MWR) GNSS receiver Doris receiver
Observation data	103 Gbit/orbit- 300 Mbit/s PLTM downlink data flow

2.2 SENTINEL-3 SPACEWIRE ARCHITECTURE

The payload architecture is built-up over a SpW network for direct collection of high rate SLSTR, OLCI and SRAL instrument's and indirect collection of low rate MWR, GNSS and DORIS instrument's plus house-keeping data through the Mil-Std-1553 bus by the SMU, all data being acquired from SpW links and managed by PDHU.

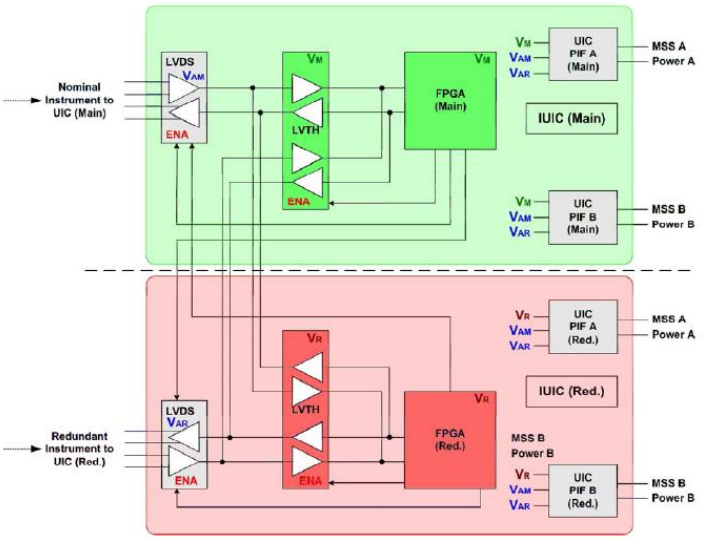


The mission data budget is easily accommodated thanks to SpW performance. Each SpW link being dedicated to point-to-point communication without interaction on the other links (no routing), the frequency is set according to the need plus a significant margin. The PDHU is able to handle the 4 SpW sources at up to 100Mb/s.

For a robust payload data management, redundancy is required. Full cross-strapping of all SpW links was abandoned for reducing harness mass, instrument's complexity and increasing the reliability vs failure propagation.

All mission data sources (OLCI, SLSTR, SRAL and SMU) provide data through two cold redundant interfaces and harnesses. The PDHU being critical as the central point of the mission data management, implements a full cross-strapping between nominal and redundant sources interfaces and its nominal and redundant sides.

SENTINEL Mission data budget		
Mission data source	continuous data rate	Link data rate
OLCI	31 Mb/s	100Mb/s
SRAL	14 Mb/s	50Mb/s
SLSTR	5 Mb/s	50Mb/s
MWR, GNSS, DORIS, HK	200Kb/s	10Mb/s



Specific efforts in design activities were spent to implement efficiently a full cross-strap redundancy within PDHU : suitable protections were implemented to prevent any risk of failure propagation from one interface to the others.

The PDHU SpW interfaces are performed thanks to a specific FPGA, instrument's ones are based on ESA Atmel SMCS-332 and SMU ones is implemented by an EPICA ASIC circuit developed by Thales Alenia Space.

The SpW full duplex capability is not used neither for synchronization nor for command/control since there is no PDHU routing capability between instrument's and SMU :

- instrument's synchronisation is ensured by basic OBT broadcast over Mil-Std-1553 bus associated to the occurrence of a PPS pulse.
- all command/control of the payload is performed through dedicated Mil-Std-1553 communication bus.

All mission data are PUS formatted at source level and transferred over SpW without additional transport protocol, i.e. one SpW cargo carrying one PUS packet : transport protocol is not needed for such point to point communication links.

2.3 SENTINEL-3 SPACEWIRE BACKGROUND

The previous similar payload architecture were based on unidirectional serial links as space LVDS interface or LNR (French acronym for fast digital line) which requires a

qualification process of commercial components. Thanks to standardized SpW links, design is simplified since development of source and destination sides can be developed in parallel with low constraints : different circuit may be implemented as SMSC-332 for instruments, FPGA for PDHU and EPICA ASIC for SMU. It also allows to tune data-rate without putting into question qualification.

On other ends difficulties were encountered in implementing SpW for :

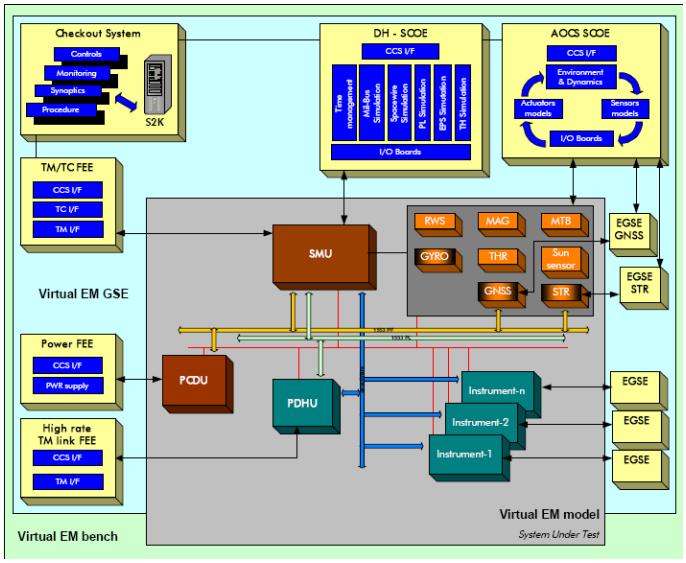
- robust cross-strapping redundancy
- harness and connector bracket impact on performance
- detector's data acquisition vs noise immunity

The two first critical points are also encountered for alternative LVDS or NLR design. The following preventive actions are performed :

- concentrate redundancy cross-strapping at PDHU, with intensive design activities to implement suitable protection against failure propagation
- plan advanced bread-boarding as part of virtual EM satellite test campaign for pre-validation of SpW communication

The last point concerning detector noise is due to the asynchronism of the SpW which prevents to synchronize the data transfer outside the optical signal acquisition slot : to prevent such perturbation risk on detector signal quality, a synchronized acquisition through a parallel bus is more suitable for detector's links.

An other advantage of using standardized SpW is found in EGSE development which is easier and cheaper thanks to existing SpW building blocks for data acquisition, simulation and investigation.



The SpW routing capability is replaced by data multiplexing when pick up from various large memory buffers before formatting, coding, encrypting for downlink to the ground. This missing routing capability prevents smooth transition for both payload synchronization and command/control allowed by SpW full-duplex, since SMU and instrument's cannot communicate directly through the SpW network.

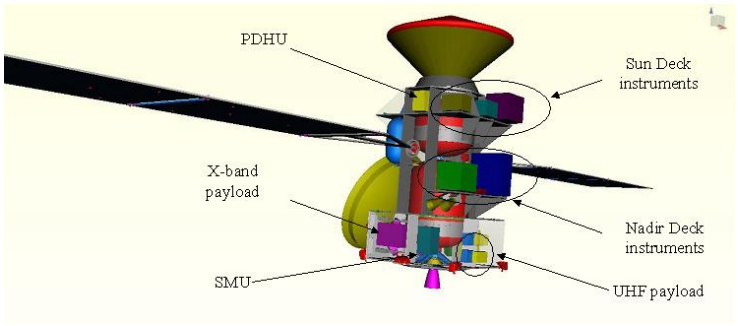
3 INTER-PLANETARY EXPLORATION CARRIER WITH SPACEWIRE

The inter-planetary missions can present an interesting target for SpW implementation due to the advantages of lower consumption w.r.t. Mil-Std-1553 bus, lower mass and flexibility in accommodating several instruments on a unified payload network for both mission data-handling and command/control.

Today the SpW implementation in inter-planetary missions is taking benefit of heritage from in-going developments, with a basic use as for sentinel missions.

3.1 EXOMARS MISSION

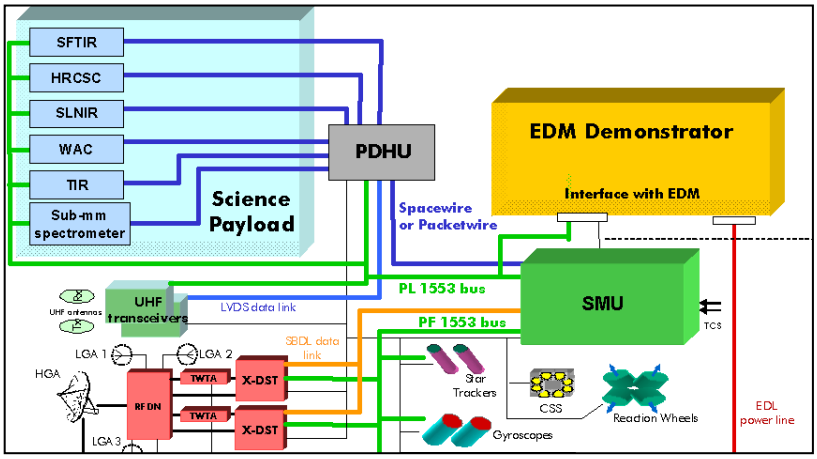
The ExoMars mission shall accomplish the technological objective with Entry, Descent and landing Module (EDM) of a payload on the surface of Mars, the scientific objective to support the search and localization of Methane sources on Mars and the data relay with Rover on Mars.



The payload sources are the UHF source with the EDL for proximity links with martian rovers, and the 6 science instruments that remotely sense the Martian atmosphere and surface.

3.2 EXOMARS SPACEWIRE ARCHITECTURE

A SpaceWire network is candidate to acquire and multiplex data from these various instruments. The payload network accommodates 6 instruments and a UHF tranceiver as proximity data link with martian rovers for their command/control messages.



Science instruments provide various data flows from 25Kb/s to 90Mb/s. The global science data volume is estimated to be lower than 15Gb per day. Science data are stored into the PDHU Mass Memory.

The payload network built around the PDHU ensures communication with 8 functional nodes: 6 instruments, the UHF transceiver and the SMU. For consumption and mass constraints, the cold redundant pair of SpW links is foreseen with a full cross-strapping redundancy implemented in the PDHU.

3.3 EXOMARS SPACEWIRE SPECIFITIES

The SpW bring the great advantage to easily accommodate instrument's with fluctuating data-rates.

Due to heritage from sentinel development, synchronization and data multiplexing are not foreseen using SpW time-codes and routing capability. A further improvement with routers could save mass and harness with a unified PL network for both synchronization, mission data and command/control.

4 GEO OBSERVATION WITH SPACEWIRE

Observation satellites from the geostationary orbit are characterized by implementing high rate instrument's with continuous mission data transfer to the ground, thanks to the constant ground station visibility. They require real time system with high throughput without any risk of bottleneck since there is no on-board storage.

These kind of missions are, up to now, dedicated to meteorological missions. Thales Alenia Space is prime contractor of meteosat satellites for more than 30 years, preparing now the third generation of meteosat satellites.

4.1 MTG MISSION

The MTG system will provide Europe's National Meteorological Services, with improved imaging and new infrared sounding capabilities for both meteorological and climate applications. The MTG space segment is based on 6 geostationary satellites carrying complementary payloads, built around a fast SpaceWire network for an unified mission and configuration data management, merging science and RF data, with more than 300Mbps continuous downlink.

4.2 MTG SPACEWIRE ARCHITECTURE

The MTG satellites accommodate respectively the FCI imager, LI imager and the DCP digital transponder for the imager S/C, and the IRS and UVN sounders for the sounder S/C, over a payload SpW network for mission data distribution and instrument's configuration with a total high rate telemetry of respectively 295Mb/s and 557Mb/s after RS concatenated encoding and encryption.

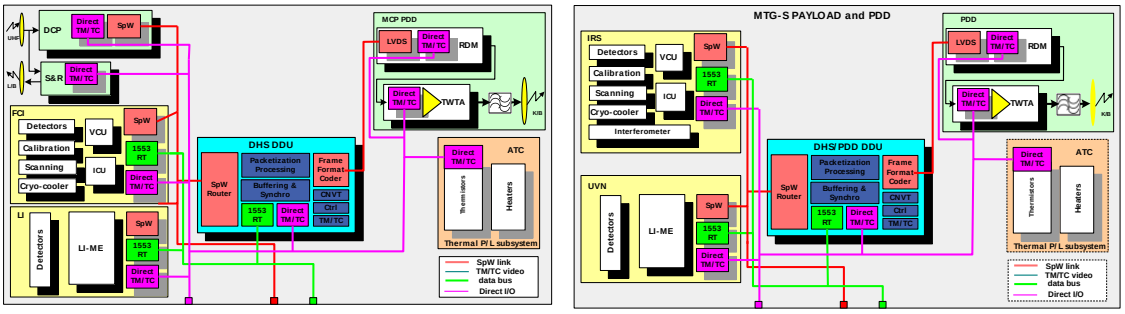
MTG Imager Mission data budget

Mission data source	continuous data rate	Total flow coded encrypted
FCI	64 Mb/s	295 Mb/s
LI	4 Mb/s	
DCP	44 Mb/s	
HK/INR	< 1Mb/s	

MTG Sounder Mission data budget

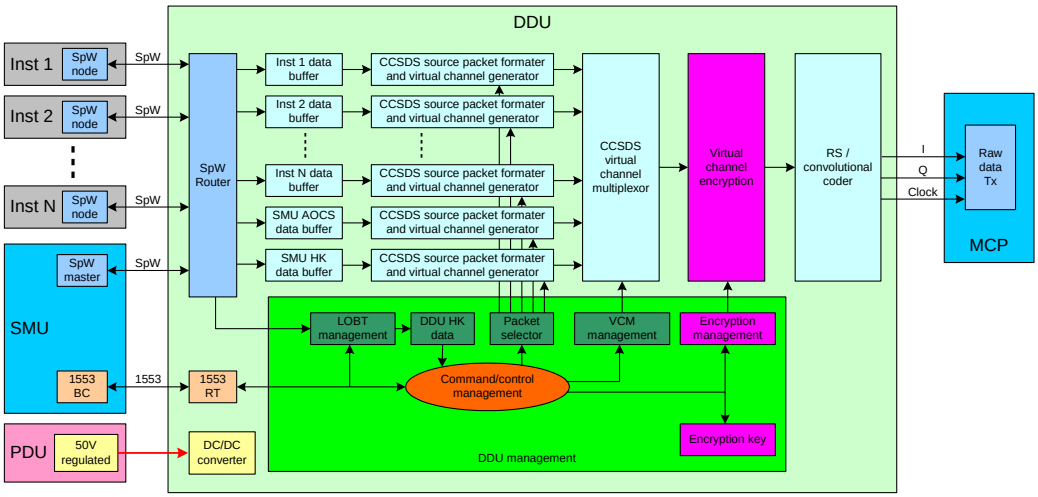
Mission data source	continuous data rate	Total flow coded encrypted
IRS1	91 Mb/s	557 Mb/s
IRS2	93 Mb/s	
UVN	54 Mb/s	
HK/INR	< 1Mb/s	

The payload data network is built around a Data Distribution Unit DDU that implements SpW routers for 3 instruments (FCI, LI, DCP or IRS1, IRS2, UVN) communication and one SMU for INR auxiliary data collection and network management. The network supports full cross-strapping between each terminal (instrument's and SMU) and DDU leading to a total of 16 terminal ports based on to independent nominal and redundant DDU SpW-10X routers.



The network architecture is identical for both imager and souder configurations. The network is running at 200Mb/s on all links providing large margins. The large margin vs data distribution and the asynchronous behaviour of the SpW link, allow to accommodate variable instrument's data flow according to their operational modes. For example the UVN instrument provides 40Mb/s in normal mode or 125Mb/s in commissioning mode.

The SpW time code distribution is not used for payload synchronization due to the instrument heritage : a classical OBT associated to a PPS pulse is broadcast through the payload Mil-Std-1553 command control bus.



Thanks to the implementation of SpW routers, the full-duplex capability of the SpW is used for command/control messages required to configure quickly instrument's without outage : large configuration tables are loaded from SMU mass memory into the instrument's through SpW links. i.e. 135Mb data transfert between 2 consecutive image acquisitions.

All messages are formatted with ECSS Packet Utilization Standard and distributed over the SpW network with ECSS SpW CCSDS transfer protocol, using user field for identifying the virtual channel for the high telemetry destination.

4.3 MTG SPACEWIRE SPECIFITIES

The MTG satellites will be the first space mission in Thales Alenia Space for implementing and using complete SpW network capability with a full cross-strapping redundancy with SpW-10X routers and full duplex used for command/control configuration messages with large tables.

Thanks to SpW standardization of physical layer and transport protocols, the PL network architecture, development and validation is simplified with well defined interfaces between the 3 instrument's and Payload Data Downlink contractors.

Similar payload were studied for the japanese satellites Himawary 8 and 9, for which Thales Alenia Space proposed to use the SpW full duplex capability to implement an active motion compensation of an US imager : the platform provided gyroscopic data at 100Hz through the SpW network to the instrument, in order to compensate by the scan mechanism attitude error for a very accurate and stable imager pointing.

5 CONCLUSION ON SPACEWIRE IMPLEMENTATIONS

As shown above, with Thales Alenia Space missions using SpaceWire (more than 12 satellites), the SpW is more and more implemented with increasing functional perimeter from mission data distribution up to configuration.

As first outcomes, implementing SpW allows to reduce interface complexity and separate interface management and development between different contractors for each side of the communication. This advantage was not possible using high speed specific line as LNR which required qualification process and common procurement.

Availability of qualified SpW tools allows to easily build EGSE and check functional behaviour.

System design effort shall be also spent at beginning of the project to define suitable redundancy concept with adequate protection preventing failure propagation as part of general construction and design specification.

The main encountered drawbacks are the procurement delays of SpW components and the lack of qualification tables or characterization tools for actual performance with final harness configuration : this is also applicable with alternative fast links as LNR.

The SpW is also analyzed in R&T and advance projects. In particular it was found interesting for IXO mission to cope with high rate data transfer over long distance. The low SpW consumption was also found attractive for new avionics sensors as far as its harness weight could be mitigated.

In next future, the use of SpW could be extended for payload command/control bringing interface and harness optimization, as soon as the determinism and FDIR for command/control messages distribution is ensured either by an efficient protocol or by architecture design rules. Then when real time performance and reliability are granted, it would also allows some interesting improvements in avionics area mainly for AOCS performance and extended operability : management of high throughput sensors and involvement of instrument's in the AOCS control closed loop for accurate satellite pointing and active motion real time compensation.

Missions and Applications 2

USING SPACEWIRE IN A SECURELY PARTITIONED COMPUTING ARCHITECTURE

Session: SpaceWire Missions and Applications

Long Paper

Peter Mendham

SciSys UK Ltd, Clothier Road, Bristol, BS4 5SS, UK

Knut Eckstein, James Windsor

ESA/ESTEC, 2200 AG Noordwijk ZH, The Netherlands

E-mail: peter.mendham@scisys.co.uk, knut.eckstein@esa.int, james.windsor@esa.int

ABSTRACT

SciSys is leading an ongoing ESA study into the development of an embedded systems software architecture which provides the capability to partition multiple applications in a safe and secure manner. This architecture targets future dual-use spacecraft shared by multiple payload developers and operators. We consider the requirements placed on a SpaceWire onboard communications system by such an architecture in terms of spatial partitioning of data and temporal partitioning of shared resources such as communications links. The resulting discussion elicits concrete requirements on both the hardware and software of the onboard SpaceWire elements.

1 INTRODUCTION

SciSys is leading an ongoing ESA study into the development of an embedded systems software architecture which provides the capability to partition multiple applications in a safe and secure manner. As a baseline reference architecture, the study considers a dual use spacecraft with a single onboard computer handling both platform and payload operations with the latter performed by separate partitioned applications.

Such a securely partitioning architecture places a number of requirements on an onboard communications system in terms confidentiality and integrity of data transmitted on shared resources such as communications links. As an increasingly popular communications technology, the applicability of SpaceWire to such a future system architecture is crucial, and the ability of a SpaceWire communications architecture to meet the requirements is the focus of this paper.

Section 2 introduces the concept of *Time and Space Partitioning* (TSP) and its applications to onboard processing. Whilst TSP has typically been applied for reasons of safety and ease of development and integration, we discuss the implications of applying TSP in an environment where security needs must also be met. The following three sections elicit concrete requirements on both SpaceWire hardware and software. In Section 3, a hardware-focussed analysis lists mechanisms by which

current SpaceWire hardware interfaces may be utilised by a securely partitioned computing platform, and presents key concepts for future consideration. Link- and network-level partitioning in routers is considered in section 4, for example time-scheduling, such as in SpaceWire-(R)T. In Section 5, a complementary analysis discusses the role of software services, such as SOIS, in a securely partitioned system. The paper concludes by summarising the central role that SpaceWire can play in a securely partitioned spacecraft architecture.

2 TIME AND SPACE PARTITIONING FOR SECURITY AND SAFETY

The concept of TSP in software systems has been established for some time, and has seen successful application in a wide variety of domains, including avionics, automotive systems, enterprise servers and handheld mobile devices.

2.1 TIME AND SPACE PARTITIONING

TSP is a technique which permits the sharing of a computing platform between multiple independent applications. *Spatial partitioning* indicates the division of shared resources, such as memory, which may be utilised by multiple applications simultaneously. Spatial regions such as memory address ranges can be limited to exclusive access by one application, or shared access by multiple applications can be granted. *Temporal partitioning* indicates the division of shared resources, such as a simple processor, which cannot be utilised by multiple applications simultaneously. Such resources must be wholly 'owned' by a single application at any point in time, and are shared by multiplexing the access to the resource by applications in time.

Through suitable enforcement of temporal and spatial partition boundaries, TSP provides an integrated environment in which applications cannot interfere with each other. This can be used for many purposes, including the isolation of sensitive applications, and permitting applications to be developed, validated and potentially certified separately. A partitioned system is shown in Figure 1.

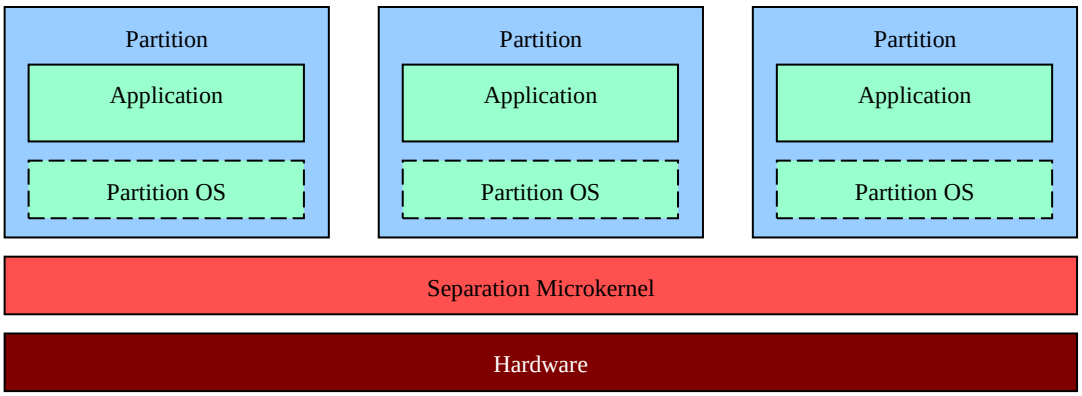


Figure 1: TSP Architecture

TSP is typically implemented using tight coordination between operating system (OS) and hardware features. A memory management unit (MMU) gives the OS an efficient mechanism to enforce spatial partitioning of mapped memory. A MMU may be used to protect memory regions from undesired read or write operations by applications which should not have access. Temporal partitioning is typically enforced through a

periodic timer interrupt which cannot be blocked or intercepted by applications. The interrupt is handled by the OS, which is responsible for saving the state of temporally partitioned resources and preparing them for access by another application.

The Integrated Modular Avionics (IMA) architecture [1], deployed on recently developed commercial aircraft such as B787 and A380, heavily utilizes TSP. In more traditional avionics architectures each element of the system is provided on a number of dedicated hardware units. In contrast, IMA permits accommodation of the computing functions of several system elements in common computing hardware and the use of shared communication networks. Typically this reduces the weight of the system and costs in development, system supply and maintenance.

The space industry has a number of similar requirements to both the aviation and automotive industries, which has lead to the investigation of the potential for "spinning-in" IMA technology, using TSP, into the space domain with studies such as the ESA IMA for Space activity [2].

2.2 PARTITIONING FOR SECURITY

As part of the ongoing Securely Partitioning Spacecraft Computing Resources activity (ESA Contract No. 22186/09/NL/LvH) led by SciSys, a study was conducted into the future security needs for primes and agencies. The study identified a wide range of scenarios in which security needs may affect onboard computing resources, such as multi-agency spacecraft, dual-use missions and assisting spacecraft manufacturers in meeting export regulations. In the reference architecture, a spacecraft features two imaging payloads: one standard resolution, to which access is not restricted; and one very high resolution payload, to which access must be restricted for commercial, legal or national security reasons. To ensure that the security needs for such a mission are met, the onboard systems must be able to guarantee confidentiality and integrity of information relating at different security levels. To reliably ensure that these security needs on a computing platform, the system designer has a number of options:

- A so-called "system-high" approach, operating the entire spacecraft, including all software applications, at the highest level of security required. This has major implications on the cost of system development and validation.

- Separate the elements of the system which correspond to different security levels onto separate hardware, limiting their interaction and adding a mass, power and volume penalties.

- Permit applications handling information at different security levels to be combined on a single computing platform by associating security information with each system entity (applications, devices etc.) and controlling access through security policies and mechanisms implemented in the operating system.

In the third approach presented above, the resulting multi-level security (MLS) operating system is typically large and complex, preventing assurance of its operation. A solution to this problem is to require the operating system to provide only reliable separation mechanisms; security levels and policy then become issues for applications and the operating system becomes simpler.

This solution is known as Multiple Independent Levels of Security (MILS) [3] and is, in many ways, similar to the TSP used in systems such as IMA, with the addition of security requirements. A system taking a MILS approach will be structured like that shown in Figure 1, with a small operating system, the separation microkernel, enforcing time and space partitioning, and applications in partitions, which may have their own operating systems.

This architecture shares many features with those of hypervisors and virtual machine monitors. The partition applications should be unaware of the other partitions on the system; in theory it should not matter if two partitions are executed on the same, or different, hardware platforms from each other. In a system where the largest concern is safety this separation is enforced to ensure a partition, either through normal operation or by malfunction, can not unintentionally influence the integrity of another partition. In a secure system this concept is extended to enforce both the integrity and confidentiality domains for the partitioned applications, i.e. it must not be possible to transfer information between two partitions, except where explicitly permitted by security policy. At the same time, the spectrum of statistical threats of malfunction and environmental influence is enlarged by the presence of a qualified, malicious human attacker. In the context of confidentiality, a *covert channel* is defined as any communication between partitions that is in contravention of the the system security policy. Two types of covert channels are typically considered: a *storage channel* involves the modification of a shared object, the state of which is used to transfer information; whereas a *timing channel* involves affecting the relative timing of observable events, such as the observation of the storage or timing events of one partition by another, which is used to impart information.

2.3 SpaceWire in a TSP System

Clearly, an onboard computer does not exist in isolation: it must interface to other onboard devices in order to receive inputs and produce outputs. SpaceWire is growing in popularity as an onboard communications medium, and is increasingly being used to interface onboard computers to both payload and platform devices. The effects of introducing SpaceWire into a securely partitioned system (i.e. one using secure TSP) fall into three categories:

- the interface between the onboard computer and the SpaceWire network, this may be part of a System-on-Chip (SoC) or it may be a separate device;
- the SpaceWire network itself, and what can be done to avoid the costs of creating two, or more, independent networks for different security levels;
- the communications software architecture, executing on the onboard computer, which is used by applications to interact with SpaceWire devices.

These topics are addressed in the following sections. As will be shown, SpaceWire is well suited for use in a securely partitioned architecture: a routed network is inherently more flexible and easier to secure than a shared medium bus such as MIL-STD-1553B or CAN.

3 SECURELY PARTITIONING THE SPACEWIRE INTERFACE

Where the SpaceWire network interfaces to the onboard processor, consideration must be given to the principles of TSP. Such considerations impact the way in which the SpaceWire interface(s) is/are to be connected to the processor, affecting address decoding and the use of interrupts and DMA.

As mentioned above, the MMU is the typical mechanism for enforcing spatial partitioning. In order to control access to interfaces, such as SpaceWire, all parts of the interface must be memory mapped. The use of alternate address spaces which are not controlled by the MMU, such as dedicated I/O spaces, should be avoided. Although the use of such spaces is typically restricted to a privileged processor mode, this requires the operating system to be involved in I/O transactions, as no application is permitted to run in a privileged mode, and increases its complexity. Where a system has multiple SpaceWire interfaces, it may be advantageous to permit these interfaces to be used by individual partitions, independently from one another. This requires that the memory-mapped resources associated with each interface are distinct (i.e. no shared registers) and that they fall into separate memory pages, so that access may be controlled with the MMU. The use of an MMU implies that an application sees only virtual, rather than physical, addresses. An interface, such as SpaceWire, however, will see physical addresses if it attempts a Direct Memory Access (DMA) transaction. Additionally, a contiguous region of virtual memory does not necessarily ensure a contiguous region of physical memory: it may therefore be difficult for a partition application to set up and manage DMA transactions without considerable operating system assistance. One way to address these issues is to introduce an I/O MMU, mapping virtual to physical addresses for devices.

Although these techniques ensure spatial separation, the use of DMA causes issues with temporal partitioning. A DMA transaction claims use of the memory bus, and access to the memory, which prevent access by the processor. Although the processor may be executing out of cache this cannot be guaranteed. Should a DMA transaction on behalf of one partition occur during the processor time allocated to a second partition, this may affect the safe operation of the second partition and also permit the second partition to observe one part of the first partition's operation. Without special provisions, therefore, DMA is not a safe technique in a TSP system, whether or not security is a concern. One way to permit the use of DMA is to utilise dedicated, dual-ported, DMA regions for each interface (see Figure 2). Access to these regions by the SpaceWire interface is largely independent to that of the processor and processor timing is not affected by DMA accesses. Buffer management operations, such as updating read and write pointers, need to be designed to permit concurrent access.

Just as an asynchronous (as far as processor execution is concerned) DMA transaction adversely affects temporal partitioning, so does the use of interrupts. In a similar manner to DMA transactions, the occurrence of an interrupt will change the execution flow of the processor and the timing of an executing partition. If hardware buffers are appropriately sized a minimum interrupt inter-arrival time can usually be assumed and with a suitable scheduling scheme, applications may meet their real time deadlines. However, such impacts on predictable partitioning are likely to be observable from unrelated partitions, creating an obvious covert channel. The only reliable way to mitigate this risk is to use polling to service interfaces.

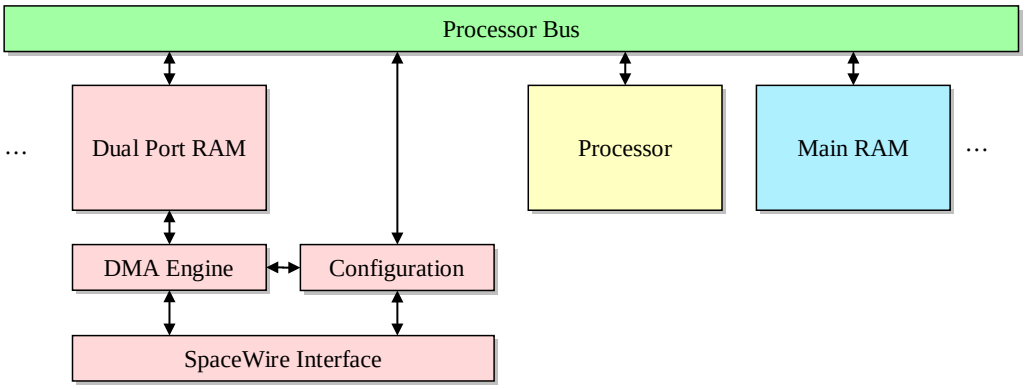


Figure 2: SpaceWire Interface using DMA on Dual-Port RAM

In the case of a SpaceWire interface, providing FIFOs are sufficiently sized for link transmit and receive rates, it should be possible to handle data transfers using polling. The handling of time-codes is potentially more complicated, and depends on the interpretation of time-codes in the system. If the arrival time of a time-code is critical, for example to update a local time counter, the handling of this may be best done without software intervention, as waiting for a polled response may introduce an unacceptable delay.

4 SECURELY PARTITIONING THE SPACEWIRE NETWORK

The most obvious way to separate SpaceWire resources corresponding to different security levels is to construct several distinct SpaceWire networks. These networks can be accessed from separate interfaces at the processor, using the guidance above. However, if these separate networks are any more that single links, this technique involves the duplication of routing resources, potentially resulting in mass, power and complexity penalties.

One way to safely partition a SpaceWire network is to restrict the destination address at the head of packets transmitted through an interface. Such a restriction may be implemented in either hardware or trusted software and involves only the first byte of the packet. Firstly, by restricting packets to use only logical addressing, it can be assured that the interface may not address packets to the configuration port of a router, or any other device. This ensures the safety and security of a configured network. Secondly, by restricting the logical addresses to which an interface may address packets, the network may be spatially partitioned along lines of function, security, or whatever suits the mission. This scheme directly associates logical addresses with security domains, and may require multiple logical addresses per node. This argument holds as long as logical addresses are never deleted by a router, so-called *regional logical addressing*. If this is not the case, and logical addresses are deleted, a further check must either be made at the interface (for the second logical address) or, more elegantly, at the entry point to this new region. The only interfaces that may configure the network are those permitted to use path addressing, which, having unrestricted access to all devices, must be fully trusted. The scheme outlined here permits the strict spatial partitioning of a SpaceWire network, requiring changes only to interface hardware and/or software, with no changes necessary to routers.

A spatially partitioned network is sufficient providing that no resources, i.e. devices or links, are shared between partitions. As routers are generally full crossbar devices, they may be shared between partitions without compromise. Where resources are shared, these must be temporally partitioned. As with the partitioning of processor execution time, to ensure both safety and security this network bandwidth partitioning must be *deterministic*.

Recent work has seen a number of proposals for the introduction of deterministic and/or low-latency traffic on SpaceWire networks, with the aim of handling time-critical command and control data. The SpaceWire-RT [4], SpaceWire-T [5] and SpaceWire-D [6] protocol proposals all share a common approach in that network bandwidth is time division multiplexed, with SpaceWire time-codes indicating the boundaries between time slots. This technique is deterministic and requires no change to network routers for its operation. Time slots are pre-allocated to interfaces by the system designer, such that the available bandwidth of any shared resource, including those shared between multiple interfaces, is not exceeded. As with the spatial partitioning case, this primarily relies on trusted network interfaces on all devices. The handling of non-trusted devices would require the enforcement of the network schedule either between device and router, or within a modified router.

Another proposal for the division of network bandwidth, utilises *virtual channels* to multiplex many channels of traffic over a single link, and by extension, an entire network [7]. The bandwidth of a link is given to whichever virtual channel has data to send and is of the highest priority. Low latency is assured by permitting high priority traffic to pre-empt low priority traffic. Whilst this scheme does successfully permit ad-hoc low latency traffic, without careful design low-priority traffic can be starved from a network. Furthermore, for secure partitioning purposes, the delivery of traffic of any but the highest priority is non-deterministic. This non-determinism may be accounted for in terms of system safety, but gives rise to the potential for covert channels, the risk of which is application dependent. The use of virtual channels, then, may not be sufficient on its own, requiring the introduction of time division multiplexing to ensure fairness and determinism. In this case the various virtual channels need not be treated hierarchically, but could be scheduled, for example, cyclically at routers. Whilst elegant and guaranteeing temporal partitioning, such a technique would require wholesale changes to both interfaces and routers.

As can be seen from the discussion in this section, SpaceWire routing resources form a critical part of a network. Existing routers may be utilised through the introduction of trusted software and/or hardware at every interface. This assumes that the target of the security effort is the TSP system, and that the potential complexity of such a trusted interface is sufficiently low to permit security assurance. If the network itself is to be secured against un-trusted devices, logical address and time slot verification functions would have to be incorporated into routers.

Trust in routing resources is key to the operation of a securely partitioned SpaceWire network. The configuration of routers has the potential to enforce spatial and, perhaps in the future, temporal partitioning. However, during the power up of a router, before it has been configured, it is potentially vulnerable to configuration by an un-trusted device. One way to prevent this issue is to associate a small configuration ROM with each router, from which a failsafe configuration may be loaded. This would “lock

down" the network into a safe and secure state, pending configuration from a trusted source.

5 SECURELY PARTITIONING ONBOARD COMMUNICATIONS SOFTWARE

The previous sections have left open the issue of where, in a partitioned architecture, the software stack responsible for communications should be placed. In Section 3, the possibility for assigning SpaceWire interfaces to partitions was discussed; however, in Section 4, the introduction of sanity checks on destination logical addresses could require trusted interface software. This indicates a trade-off between a decentralised approach, in which the operations of partitions are self-contained and may be validated separately; and a more centralised approach in which a single, trusted, partition is responsible for managing the SpaceWire interfaces.

The latter approach was taken by SciSys in the Payloads with Resource-efficient Integration for Science Missions (PRISM) project. Here, the RTEMS operating system was modified to include temporal partitions, including the enforcement of both processor execution time and I/O bandwidth budgets. Including the concept of I/O partitioning in the system architecture permitted the I/O handling software to be located in a single system partition, and shared between other applications. Whilst a promising approach for integration and increased safety, the PRISM operating system is relatively complex and would be difficult to assure for security purposes.

The best way to reduce the potential assurance effort is to minimise the size of the trusted component. A good example is the application of the CCSDS SOIS stack []. Here, the SpaceWire subnetwork services could be associated with an individual interface, perhaps in a trusted partition, whereas the application support services would be placed in one or more un-trusted partitions. Especially where there is good hardware support, the SpaceWire subnetwork services may be relatively simple, increasing the potential for assurance. The SOIS architecture provides a good template for the decentralisation/centralisation split in the onboard communications architecture.

6 CONCLUSIONS

6.1 REQUIREMENTS FOR THE USE OF SPACEWIRE IN A SECURELY PARTITIONED ARCHITECTURE

As a routed network of flexible topology, SpaceWire has the potential to integrate well into a securely partitioned system. However, there are a number of challenges when applying secure TSP to SpaceWire: some are specific to SpaceWire and others less so.

At the interface level, consideration should be given to the potential for spatially protecting interfaces using the MMU. This requires memory-mapped hardware, laid out in consideration for page boundaries. Shared resources, such as the processor, must be deterministically shared in time and therefore cannot be disrupted by asynchronous events such as DMA or interrupts. The design of SpaceWire interfaces and driver software should account for this.

At the network level, it is possible to spatially partition a SpaceWire network using logical address verification at interfaces and careful network design. Deterministic temporal partitioning, such as that employed in SpaceWire-(R)T and SpaceWire-D, meets the requirements for TSP; whereas the use of virtual channels may not when applied alone. With the addition of time-division multiplexing it becomes more powerful. Simple network TSP can be introduced using trusted hardware/software at interfaces, and is sufficient if the focus of security requirements is a software TSP system. The introduction of un-trusted devices on a SpaceWire network necessitates modifications to router technology. To ensure router enforcement of partitioning, routers should ideally use a configuration ROM to ensure a trusted boot.

At the software level, where an interface must be trusted, and shared between multiple partitions, the trusted code should be minimised. The subnetwork layer in SOIS is a logical point to split the communications stack.

6.2 LOOKING TO THE FUTURE

SpaceWire has the potential to play a central role in TSP architectures including securely partitioned onboard systems, providing that consideration is given to the topics presented in this paper. Some issues, such as those at the interface level, can be addressed in the short term, either by use of existing devices or minor modification to current IP. A SpaceWire network can be securely partitioned using current technology, providing all node interfaces are trusted. Relaxing this caveat requires modification to routers: as the arbiter of network traffic, routers are the appropriate place for the secure control of network bandwidth. For the current generation of SpaceWire devices such changes would be an addition or modification to standard behaviour; as thought is given to the next generation of SpaceWire technology, it is suggested that (secure) TSP should be carefully considered as it is likely to become an important technique in the design of spacecraft onboard systems.

7 REFERENCES

1. Airlines Electronic Engineering Committee, 'ARINC Specification 651: Design Guidance for Integrated Modular Avionics', 1997.

2. Integrated Modular Avionics for Space. ESA ITT AO/1-6295/09/NL/LvH.

3. Alves-Foss, J. et al, 'The MILS Architectures for High Assurance Embedded Systems"' International Journal of Embedded Systems, 2(3/4), 2006.

4. University of Dundee, "SpaceWire-RT: Initial Protocol Definition', Version 2.1, Available from the minutes of the 12th SpaceWire Working Group, ://spacewire.esa.int/WG/SpaceWire/, 2010.

5. Parkes, S., "SpaceWire-RT and SpaceWire-T', Available from the minutes of the 14th SpaceWire Working Group, ://spacewire.esa.int/WG/SpaceWire/, 2010.

6. University of Dundee, "SpaceWire-D Protocol', Available from the minutes of the 14th SpaceWire Working Group, ://spacewire.esa.int/WG/SpaceWire/, 2010.

7. Cook and Walker, 'Virtual Networks', Available from the minutes of the 13th SpaceWire Working Group, ://spacewire.esa.int/WG/SpaceWire/, 2010.

ENHANCED DYNAMIC RECONFIGURABLE PROCESSING MODULE FOR FUTURE SPACE APPLICATIONS

Session: SpaceWire Missions and Applications

Long Paper

Frank Bubenhagen, Björn Fiethe, Harald Michalik, Björn Osterloh

IDA TU Braunschweig, Hans-Sommer-Str.66, D-38106 Braunschweig, Germany

Paul Norridge, Wayne Sullivan, Chris Topping

Astrium Ltd, Gunnels Wood Road, Stevenage, Herts, UK SG1 2AS3

Jørgen Ilstad

European Space Agency, ESTEC, Keplerlaan 1, Noordwijk ZH, Netherlands

E-mail: bubenhagen@ida.ing.tu-bs.de, fiethe@ida.ing.tu-bs.de, michalik@ida.ing.tu-bs.de, b.osterloh@tu-bs.de, paul.norridge@astrium.eads.net, wayne.sullivan@astrium.eads.net, christopher.topping@astrium.eads.net, jorgen.ilstad@esa.int

ABSTRACT

Future space missions require high-performance on-board processing capabilities and a high degree of flexibility. State of the art radiation tolerant SRAM-based FPGAs with large gate count provide an attractive solution for in-flight dynamic reconfigurability. With these devices an advanced System-on-Chip (SoC) can be implemented, but also the system reliability and qualification has to be guaranteed for the harsh space environment. Therefore single modules have to be isolated from the system physically and logically by qualified communication architecture, presented in this paper: The SpaceWire based System-on-Chip Wire (SoCWire) communication network. SoCWire provides a safe way to dynamically reconfigure parts of the FPGA during flight. First verification results of a dynamic reconfigurable SoC based on SoCWire are presented. Developed around SoCWire, the basic architecture for an advanced Dynamic Reconfigurable Processing Module (DRPM) is proposed.

1 INTRODUCTION

For data processing of payload instruments on scientific spacecrafts specific processing modules are commonly used. With increased data rates and the requirement to control multiple sensors, the need for increased on-board processing capabilities and a higher degree of instrument autonomy grow. While there are higher requirements for a data processing on the one hand, on the other hand some basic conditions remain still the same, i.e. limited downlink capacity, limited resources of power and mass. Also the need for shorter development times and the demand by scientists to adapt the instrument to mission specific requirements even after launch require an advanced architecture. This leads to an in-flight adaptable hardware architecture, which guarantees the once on-ground achieved qualification even after partial exchange of hardware modules.

Today the SRAM-based Virtex FPGAs from Xilinx provides high logic capacity and thus offer a highly flexible platform to implement a reconfigurable System-on-Chip (SoC) in a single device. These devices are available in radiation tolerant versions and already have proven reliable flight heritage in many space missions, e.g. ESA Venus Express (VEX) or NASA Dawn. However, the full flexibility of these devices to perform complete or partial reconfiguration even during operation was only used throughout the development phase on ground so far.

For an enhanced reconfigurable system the system qualification has to be guaranteed. Effects during the reconfiguration process, space radiation induced errors and interference of updated modules on the system have to be prevented. Therefore, updated modules have to be isolated physically and logically by a qualified communication architecture from the system.

This paper presents the key element for such an enhanced architecture, the SpaceWire based System-on-Chip Wire (SoCWire) communication network. SoCWire provides a safe way to dynamically reconfigure parts of the FPGA during flight. First verification results of a dynamic reconfigurable SoC based on SoCWire are presented. At last the basic architecture for an advanced processing module is proposed.

2 EFFECTS WITHIN A RECONFIGURABLE FPGA

The use of Xilinx SRAM-based FPGAs for a dynamic reconfigurable system requires considering of two effects: (i) glitch effects, which occur during the dynamic partial reconfiguration process while the FPGA is in operation and (ii) SEUs (Single-Event-Upsets) within the space environment.

Partial reconfiguration denotes the modification of a limited, predefined portion of a FPGA. A minimal reconfigurable system consists of a static area, which remains unchanged and a Partial Reconfigurable Area (PRA), which is shared by two or more Partial Reconfigurable Modules (PRMs) with different functionality. Xilinx FPGAs have no explicit activation technique for a PRA. Therefore the configuration frames become active as they were written. Configuration bits remaining unchanged will not glitch during reconfiguration, but bits with a change of its logical state could momentarily glitch when the frame write is processed. Experiments with reconfiguration of a PRA from PRM1 to PRM2 and vice versa have shown unpredictable behaviour for both, the duration of glitches and their influence on the interface between the PRM and the static area.

A SEU is caused by charged particles losing energy by ionizing the medium which they pass and leaving behind electron-hole pairs. Within a memory cell or flip-flop this can cause a change of state and consequently corrupt the stored data. The configuration for the programmable elements and routing resources of a Xilinx FPGA is stored within static memory cells. Falsified memory cells can be corrected by "scrubbing", i.e. continuous reloading of configuration memory with the initial design, but this does not prevent a propagation of an error through the system. Techniques like Triple Modular Redundancy (TMR) can mitigate error propagation. The drawbacks of TMR are higher resource utilization, a decrease of speed due to longer paths and an increase of current because of more logic. Typically processing units for scientific instruments are not mission critical. As result a trade-off between limited resources and instrument availability is partly applied TMR. Anyhow, a SEU

in a non-TMR PRM interface logic could block the communication architecture and stop the system.

Taking into account glitch effects and SEU induced errors during dynamic partial reconfiguration the system qualification in a classical bus-based architecture within a FPGA cannot be guaranteed. An enhanced architecture is required, which isolates PRMs from the TMR protected host system to guarantee uninterruptable operation of the system.

3 System-on-Chip Wire (SoCWire)

SoCWire has been developed to provide a Network-on-Chip (NoC) architecture which is able to connect several PRMs with a host system and concurrently isolate the PRMs logically and physically. SEU induced error, glitch effects or an intended replacement of a module does not affect the operation of the remaining system.

3.1 SoCWire Basics

Available spacecraft communication standards, e.g. MIL-STD-1553B, CAN bus, SpaceWire were analyzed and compared for their suitability for a NoC. The outcome of this analysis was that SpaceWire as an asynchronous, point-to-point, bi-directional, serial link interface with a credit-based flow control, error detection, hot-plug ability and automatic reconnection after a link disconnection [1] is currently the only available switch topology and most suitable for a fault-tolerant and robust NoC approach. As mentioned before SpaceWire is an asynchronous interface and performance depends on skew and jitter. Reconfigurable processing modules are implemented within a complete on-chip environment (NoC approach). Therefore, the Spacewire interface has been modified to a synchronous, 10bit parallel data interface (8bit data, control flag, parity bit), which results in significantly higher data rates compared to the SpaceWire standard, e.g. 800Mbit/s at clock frequency of 100MHz. Additionally, the data word width is scalable from 8bit to 128bit, which further improves the throughput. Furthermore, the advantageous and reliable features from this standard, such as flow-control, error detection and automatic link recovery in case of an error, were preserved. Since SoCWire operates in a complete synchronous environment, the timeouts during initialization and detection and recovery after a link disconnection could be significantly decreased.

3.2 SoCWire Network

To build up a network, a switch and a packet oriented protocol is needed. A SoCWire network as shown in Figure 1 comprises: SoCWire coder/decoder (CODEC) as network interface and a SoCWire switch to route the data packets through the network [2]. The SoCWire switch is again based on the SpaceWire standard. A SoCWire CODEC connects a node or the host system typically via a SoCWire switch to a SoCWire network. The nodes are similar to SpaceWire nodes source and destination of a link. The SoCWire switch is scalable from 8bit to 128bit data word width and provides a configurable number of up to 32 ports. In contrast to a SpaceWire router the configuration port was discarded and logical addressing is not supported to save resources. A simple path addressing scheme is implemented instead, which is suitable for small on-chip networks. The SoCWire switch comprises wormhole routing and the simple time slot based round robin scheduling algorithm.

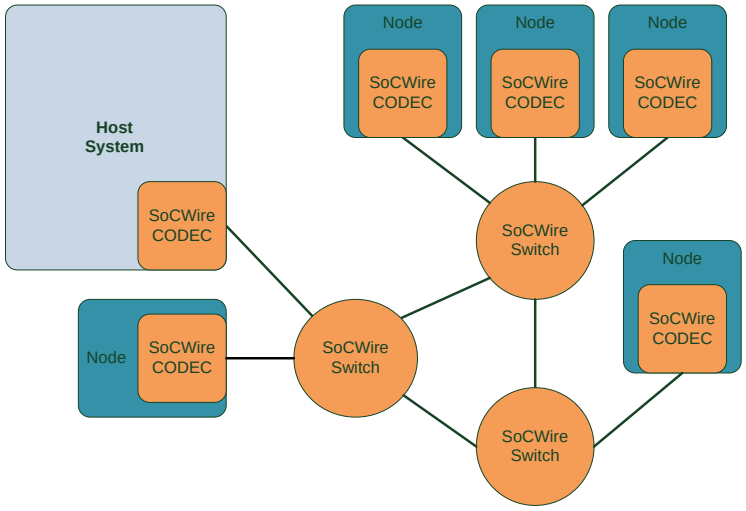


Figure 1 SoCWire architecture network example

4 SoCWire: Architecture Verification

The objective of SoCWire is to provide a robust communication architecture for dynamic partial reconfiguration systems. Since the major requirement for SoCWire is the isolation of a PRM, this feature has to be validated in an architecture verification.

4.1 Functional Verification

SoCWire has to be validated on the advantageous features of SpaceWire providing link initialization, error detection/recovery and unidirectional and bidirectional data rates. The main difference between SpaceWire and SoCWire is that SoCWire provides a parallel data interface and operates in a completely synchronous environment. The advantage of this point is that SoCWire is more deterministic, because any change of state is related to clock cycles. SoCWire is a fully pipelined implementation and two clock cycles are required to perform an action. One advantage of the synchronous environment is the much faster initialization of a link in comparison to SpaceWire. A disconnection is detected after three clock cycles, the exchange of silence lasts six clock cycles and the timeout twelve clock cycles. Overall, 26 clock cycles minimum are necessary for building up a link on the condition that both SoCWire CODECs receive the reset at the same time. Tests with adding delays of different length to one of these reset signals always resulted in a proper initialization of the link. Both the unidirectional and the bidirectional data transfer have been tested with a Pseudo Random Bit Sequence (PRBS) generator stimulus to validate data integrity. Furthermore, data packets of different length (1 to 1048576 bytes) have been tested. In all performed tests no transmission errors have been detected and the data rates from the simulations could be verified. Furthermore, SoCWire has been tested and validated on the error detection/recovery features of the SpaceWire standard, e.g. parity errors, escape errors, character sequence errors, credit errors and disconnect errors. Figure 2 depicts the fault injection mechanism for this verification. All errors have been successfully injected and error detection and recovery could be validated to be SpaceWire conform. The error recovery time of SoCWire is at minimum the initialization time for a link plus synchronization overhead.

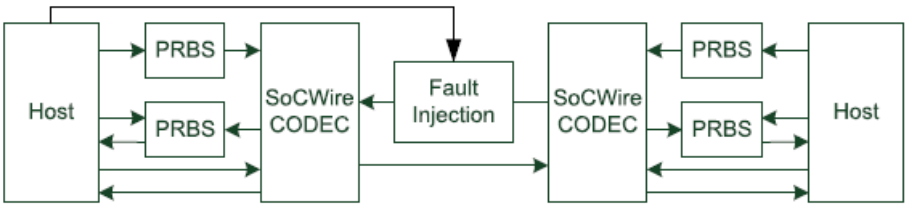


Figure 2 SoCWire verification architecture

4.2 FAULT TOLERANCE

One mandatory requirement in a space environment is fault tolerance. Single-Event-Upsets on a SoCWire node within the FPGA can be modelled as stuck bit either at logical ‘0’ or ’1’. This error can occur during a link initialization or during run-time. With the programmable fault injection mechanism a certain bit in the link has been fixed to one of the logical states. In all performed test a stuck bit either in the link initialization phase or during run-time does not affect the functionality of the host system and an error is reported. Appropriate error recovery schemes can be applied by the user, e.g. scrubbing.

During the reconfiguration process of a PRM glitch effects occur and affect nearly every part of the interface logic for a given time in the range of microseconds. These effects impact the link initialization phase when an empty PRA is configured for the first time or during an established link connection when a PRM is replaced by another one. To verify the impact of glitch effects on a SoCWire CODEC interface, a random pattern generator with random delay in the range of nanoseconds to several microseconds was implemented in hardware. This generator emulates the behaviour on the interface signals which could occur with different PRM configuration patterns. Even though this generator does not simulate the real FPGA technology and effects during dynamic partial reconfiguration, during the test the SoCWire host system was not disturbed in its operation.

4.3 PARTIAL DYNAMIC RECONFIGURATION

A dynamic partial reconfigurable SoCWire architecture with host system including SoCWire CODEC and a PRM with SoCWire CODEC as well as an additional module for control and data generation e.g. PRBS has been implemented. Moreover, a static PRM with all outputs ones and a PRM with pure counter functionality have been created. The following tests have been performed with the JTAG interface to reconfigure the system dynamically: (i) SoCWire counter module to SoCWire PRBS module, and (ii) Static module to SoCWire PRBS module. Since dynamic partial reconfiguration has the same behaviour as scrubbing on all elements within a module which does not change, test (i) was performed to prevent the SoCWire CODEC from not being affected by the dynamic reconfiguration process.

Two behaviours have been observed during the tests, which are shown in Figure 3. The figure represents the “active signals” or link connected from the SoCWire CODEC core on host system side and on PRM side. (I) shows a smooth dynamic reconfiguration of the PRM. Glitches occurred on all PRM outputs during the reconfiguration process. (II) shows the glitch effects as well as a repeatedly establishing link connection stabilising at the end. There is not much known about the

dynamic partial reconfiguration process in Xilinx FPGAs to explain this effect. Configuration frames become active when they are written; it is most likely that parts of the design operate before the reconfiguration process is finished.

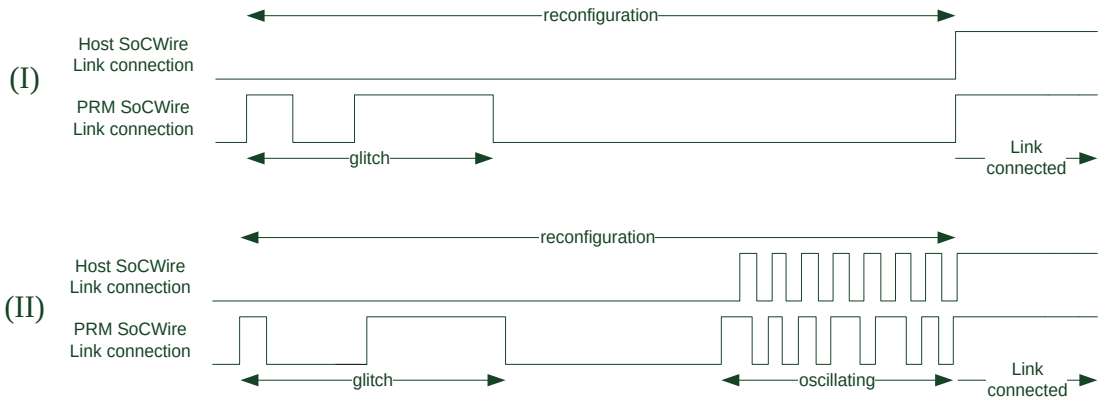


Figure 3 SoCWire "active signals" during dynamic partial reconfiguration

Since the configuration memory within a Xilinx FPGA is written from the left to right side [3], also the influence of bus macro placement, which establishes communication between static area and PRAs, has been analysed. During the initial tests the bus macros were placed on right side of the PRM as depicted in Figure 4 on the left hand side. Tests with placement of the bus macros on the right hand side of the PRM showed a smooth stable link connection avoiding the oscillation effect. Even with oscillating behaviour during the dynamic partial reconfiguration process, the PRMs were isolated from the host system and do not have any effect on its operation.

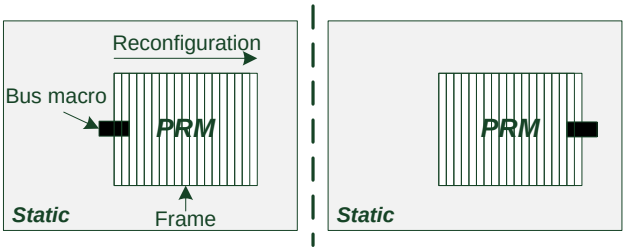


Figure 4 PRM bus macro placement

5 DRPM ARCHITECTURE

Around the SoCWire communication architecture we are currently developing under ESA contract a flexible processing system with full support for in-flight dynamic partial reconfiguration of application firmware, the Dynamic Reconfigurable Processing Module (DRPM). The basic DRPM architecture is shown in Figure 5 [4]. The major subunits are (i) dynamically reconfigurable FPGAs (within each DFPGA), (ii) SpaceWire router for hosting and managing the networking between various subunits, (iii) system controller for overall configuration control of the module and execution of application software, (iv) interfaces to spacecraft using standards like SpaceWire, MIL-1553B and CAN bus, and finally (v) interfaces to the instrument electronics, e.g. sensors or cameras.

The DRPM comprise a highly modular architecture. Consequently, the SpaceWire router can provide expandability not only to additional DFPGAs, but also to

additional DRPMs. With this concept it is possible to simply extend the processing capacity by attaching additional modules or adding modules for hardware redundancy. Then one system controller would be the master and the other ones slaves.

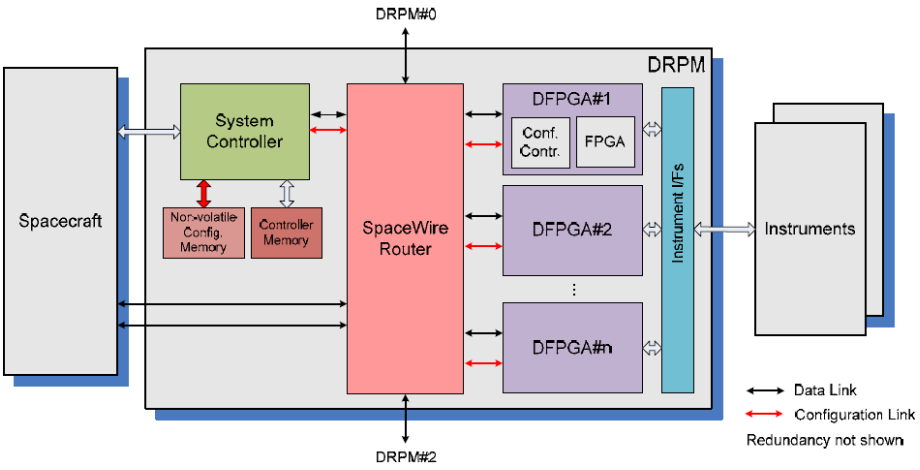


Figure 5 DRPM architecture

Since the system controllers' main task is controlling and supervising the overall DRPM, a fault-tolerant processor implementation should be used for this subunit. For instance, the LEON-based SpaceWire RTC ASIC (AT7913E) already incorporates the required interfaces like RMAP compatible SpaceWire and CAN bus controller. Of major importance is a safe and flexible implementation of the high capacity non-volatile memory for secure storage of all partial configuration bit files needed for the DFPGAs. Each reconfigurable DFPGA consists of configuration controller containing the static area with common interfaces and one or several reconfigurable FPGA(s), mainly comprising the dynamic area. This basic architecture is depicted in Figure 6.

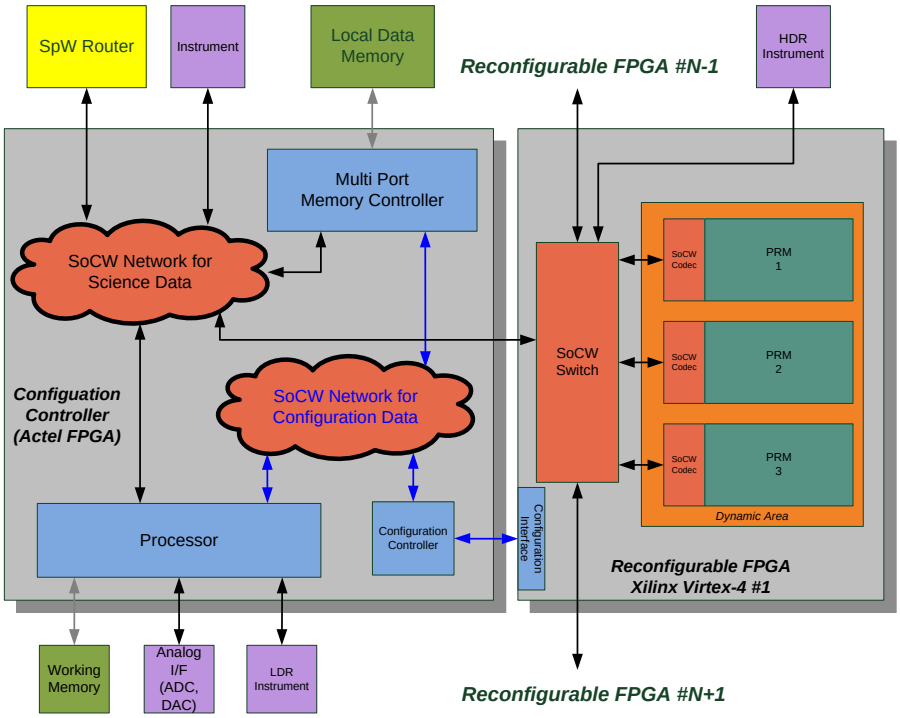


Figure 6 DFPGA architecture

The configuration controller is responsible for configuration, verification and supervision of PRMs within the dynamic area. It is implemented within a TMR by design one-time programmable RTAX FPGA. Two independent SoCWire communication networks form the backbone for controller functionality. One network is responsible for secure dynamic configuration and scrubbing of reconfigurable FPGA(s). The other one connects the dynamic area with the processor of the controller, instrument interfaces and a large local data memory. To achieve high reliability this memory is implemented in the static area with advanced symbol error correction capabilities for secure temporal storage of local configuration files. The independency of the two SoCWire networks provides increased reliability. The processor is required to provide data-flow control functions for the allocation and access of various interfaces to the commonly used data memory and configuration management of the attached reconfigurable FPGA(s). The dynamic area is based on Xilinx Virtex-4 FPGAs which are available on reliable packaging and certified for radiation performance and reliability. The SoCWire switch within the small static area of the reconfigurable FPGA(s) connects to the different PRMs and optionally directly to external high-speed interfaces, e.g. Channel Link. To achieve a modular architecture, the switch provides also direct data exchange between different reconfigurable FPGAs or even DFPGA subunits.

6 CONCLUSION

The DRPM provides an architecture being suitable to satisfy the demand of future space missions for high performance on-board processing with the requirement to update processing modules in-flight. One issue within such an enhanced architecture is the guarantee of system qualification, even after an update of a processing module. SpaceWire is widely used as a proved reliable interface standard on-board spacecrafts. Modifying this standard to the fault-tolerant, high-speed on-chip communication architecture SoCWire for FPGAs offers the possibility to built-up systems where processing modules can be exchanged without affecting the operation of the host system. SoCWire is published as an open source project provided by IDA. Source code, documentation and testbenches can be accessed at www.socwire.org.

7 REFERENCES

1. ESA ESTEC, “Space Engineering: SpaceWire-Links, nodes, routers, and networks”, ECSS-E-50-12A, Noordwijk Netherlands, January 2003.

2. B. Osterloh, “SoCWire User Manual“, www.socwire.org, 2009

3. Xilinx Inc., “Virtex-4 FPGA Configuration User Guide. UG071(v1.11)”, www.xilinx.com, USA, 2009.

4. ESA, “FPGA bases generic module and dynamic reconfigurator”, TEC-EDP/2008.30/JI, Issue: 1 Rev.1, Noordwijk, Netherlands, 2008

SPACEWIRE/RMAP-BASED DATA ACQUISITION FRAMEWORK FOR SCIENTIFIC INSTRUMENTS: OVERVIEW, APPLICATION, AND RECENT UPDATES

Session: SpaceWire Missions and Applications

Short Paper

Takayuki Yuasa, Wataru Kokuyama, Kazuo Makishima, Kazuhiro Nakazawa,
The University of Tokyo, 7-3-1 Hongo, Bunkyo, Tokyo, Japan 113-0033

Hirokazu Odaka, Motohide Kokubun, Takeshi Takashima, Tadayuki Takahashi,
Institute of Space and Astronautical Science (ISAS), Japan Aerospace Exploration Agnency (JAXA), 3-1-1 Yoshinodai, Sagamihara, Kanagawa, Japan 229-8510

Masaharu Nomachi,
Laboratory of Nuclear Studies, Graduate School of Science, Osaka University, 1-1 Machikaneyama, Toyonaka, Osaka 560-0043

Iwao Fujishiro, and Fumio Hodoshima

Shimafuji Electric Incorporated, 8-1-15 Nishikamata, Ota, Tokyo, Japan 144-0051

E-mail (TY) : yuasa@juno.phys.s.u-tokyo.ac.jp

ABSTRACT

We report a data acquisition framework for scientific instruments based on SpaceWire interfaces. The framework consists of computers and front-end electronics which have SpaceWire ports in their small footprints. RMAP protocol stack, portable class library, and a template of hardware logics are provided with these hardwares for users to implement their own applications. The framework has been employed in multiple practical developments mainly for Japanese X-ray satellite. The protocol stack was also utilized in a flight model of the Japanese small demonstration satellite which was launched in 2009, and successfully tested its SpaceWire and RMAP functionality in orbit.

1 SPACEWIRE-BASED DATA ACQUISITION FRAMEWORK

1.1 BUILDING BLOCKS

As shown in Figure 1, our data acquisition framework consists of four principal components.

Scientific instruments - output scientific data as analog and digital signals,

Front-end SpaceWire boards - receive the data and stores them into SDRAM. A typical SpaceWire board has two FPGAs on it; one is dedicated for SpaceWire/RMAP IP core, and users can implement instrument-dependent hardware logic to the other one. The SDRAM and the users FPGA can be accessed via RMAP.

SpaceWire router - connects multiple SpaceWire links constituting a network.

Read-out control computer - executes users' read-out program to transfer the data from the boards via SpaceWire/RMAP. The data can be stored on the computer itself, or on a remote PC.

In laboratory experiments (or non-flight experiments), products by Shimafuji Electric, presented in Figure 2, are utilized as these components. Shimafuji's 6 Port Router, whose size is almost the same as that of SpaceCube, became available recently. In some experiments, an 8-port router from NEC Corp has been used. Footprints of these products are fairy small compared to conventional crates and modules of VME or CAMAC systems, and SpaceWire has network capability not just a data bus. These greatly contribute to make the experimental setup compact and logically well structured. The setup is also highly scalable since we can increase the number of instruments and attached front-end circuit boards by simply introducing new routers.

The link speed of SpaceWire interfaces of the front-end SpaceWire boards and SpaceCube computer is variable, and usually we operate them at 100 MHz.

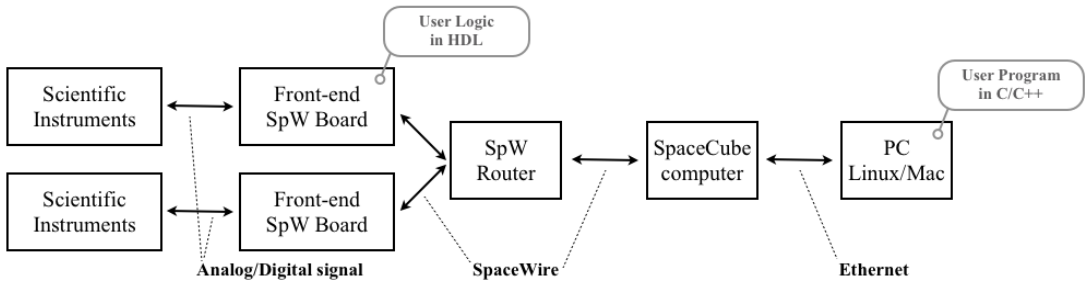


Figure 1 : Components used in the present SpaceWire-base data acquisition framework. User defined logic and program can be implemented on front-end boards and SpaceCube computer, respectively.

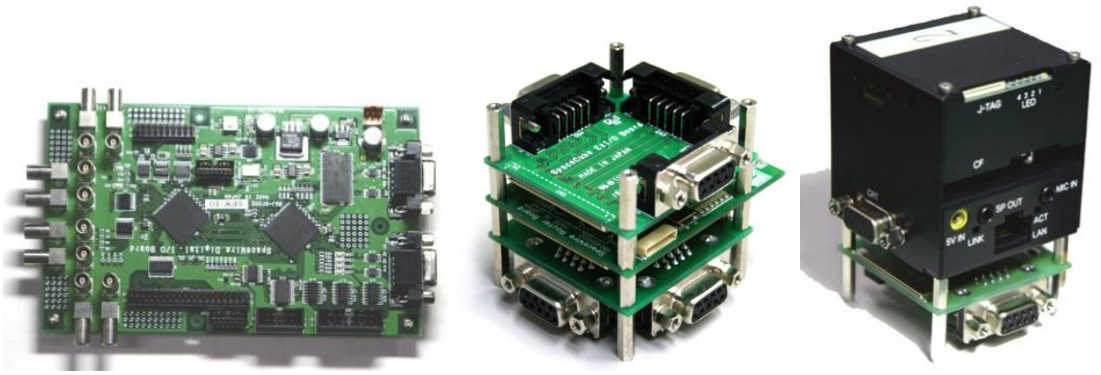


Figure 2 : An example of a front-end SpaceWire board, SpaceWire Digital IO (left), SpaceWire 6 Port Router (center), and SpaceCube computer (right) by Shimafuji Electric. For specs, see e.g. [1].

1.2 PROVIDED SOFTWARES AND THEIR FEATURES

An RMAP protocl stack and related class libraries for user programs on SpaceCube is available in C++ language (SpaceWire/RMAP Library [2]). They are designed to be portable, i.e. independent from hardware types and operating systems, and

implementations of the system dependent layers for TRON-base realtime operating system (SpaceCube) and POSIX (Linux and Macintosh) are included. Therefore, once a user writes a program along with the library, its source code can be used to produce binaries for both of SpaceCube and ordinary PCs by just re-compiling.

If there is no SpaceWire interface available on an ordinary PC, SpaceCube can be a virtual SpaceWire interface via TCP/IP (Ethernet) as presented in Figure 3. A protocol converter provided with the library should be started on SpaceCube, and it seamlessly transmits SpaceWire packets to a user program running on a PC over TCP/IP connections. The basic idea is describe in detail in [3] and references therein. With this protocol converter, a development and debugging of a user program become easier since most of work can be done on a PC. Therefore, if an additional PC (its size and cost) is acceptable in an experiment, it is worth considering to execute a user program on a PC not on SpaceCube. Since this is a software protocol converter, information of the character level cannot be reconstructed; e.g. NULL does not appear in a TCP/IP end.

We have also been providing a template VHDL files for a hardware logic for an FPGA on a front-end circuit board [1]. The template includes several typical functionalities, needed to construct instrument-dependent logics, such as an on-chip data bus, a bus adapter, and a skeleton for registers and user original modules. RMAP accesses to a front-end board are translated into local bus access to the FPGA. The FPGA, or user logic on it, can respond to an RMAP Read and Write accesses by returning data to the local bus and by updating register value with data passed via the local bus, respectively.

From 2008, the source code of the SpaceWire IP core and the RMAP IP core used in Shimafuji's products is publicly released via their web site so that users can implement it into their own circuits.

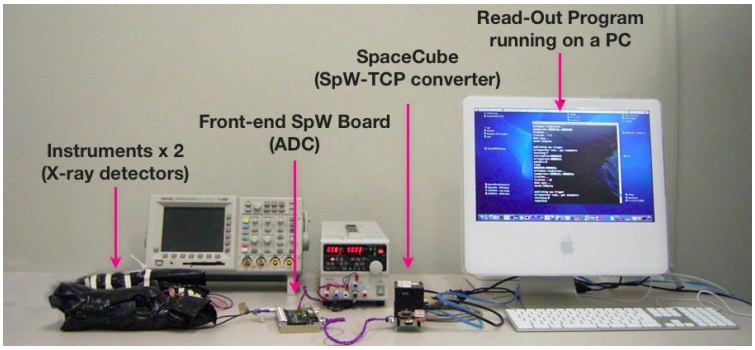


Figure 3 : An example experiment which utilizes SpaceWire-to-TCP/IP converter on SpaceCube. A user program runs on a PC, and transfers SpaceWire packets via a SpaceWire interface of SpaceCube.

1.3 PERFORMANCE

When a link is operated at 100 MHz, SpaceCube transfers SpaceWire packets at ~30 Mbps without any packet processing. If the software RMAP implementation is used in a user program on SpaceCube, data transfer rate drops to ~8 Mbps because of emerging CPU load. The SpaceWire-to-TCP/IP converter does not interpret packet content but just transfers them to/from a SpaceWire interface and a TCP socket. Therefore, in this case, the transfer speed is limited by the maximum transfer rate, ~10 Mbps, of a TCP protocol stack used in SpaceCube over its 100 Mbps Ethernet interface.

2 PRACTICAL EXAMPLES

2.1 APPLICATIONS IN GROUND EXPERIMTNS

We have been utilizing this data acquisition framework in many real experiments. Since our background is X-ray and Gamma-ray astrophysics, examples include development and test of a new X-ray microcalorimeter by JAXA, a balloon borne hard X-ray mission by Hiroshima and PoGO experiment group, and other radioactive measurements. As presented by [4,5], the framework has been put into commission in developments of scientific instruments for the Japanese next X-ray observatory satellite, ASTRO-H, which is scheduled to be launched in 2014.

2.2 TEST IN ORBIT

Japanese Small Demonstration Satellite 1 which has onboard SpaceWire Interface test Module (SWIM) together with other test components, was launched in January 2009. SWIM consists of two sub-modules flight model of SpaceCube computer (SpaceCube2 by NEC and JAXA, [6]) and a scientific instrument module SWIMμν with SpaceWire interfaces (Mitsubishi Heavy Industrial Corp, University of Tokyo, and JAXA). A subset of the RMAP protocol stack provided in the class library was implemented on SpaceCube2, and a user program developed using the protocol stack successfully performed scientific data transfer between SpaceCube2 (RMAP master) and SWIMμν module (RMAP target) in orbit. This proved that the concept of the SpaceWire data acquisition framework works well even in a flight model of a satellite.

3 FUTURE WORK

To improve the transfer speed of the SpaceWire-to-TCP/IP converter, we may have to use GbEther and a TCP/IP stack implemented on a dedicated hardware, e.g. ZestET1 by Orange Tree Tech, so as to increase the bandwidth and to reduce the CPU load for TCP. Since the development of ASTRO-H instruments are becoming very active, it is useful for the class library to support breadboard model and flight model of onboard computers.

4 REFFERENCES

1. Yuasa et al., “Development of a SpW/RMAP-based Data Acquisition Framework for Scientific Detector Applications”, SpaceWire Conference, Dundee, September 2007
2. Yuasa et al., ”A Portable SpaceWire/RMAP Class Library for Scientific Detector Read Out Systems”, SpaceWire Conference, Nara, Japan, November 2008
3. S. Mills and S. Parks, “The SpaceWire internet tunnel and the advantages it provides for spacecraft”, SpaceWire Conference, Nara, Japan, November 2008
4. T. Kouzu et al., “Verification of high resolution timing system with SpaceWire network onboard ASTRO-H”, this conference
5. T. Fujinaga et al., “Development of SpaceWire based data acquisition system for the X-ray CCD camera on board ASTRO-H”, this conference
6. T. Takahashi et al., “SpaceCube 2 -- An Onboard Computer Based on SpaceCube Architecture”, SpaceWire Conference, Dundee, September 2007

OVERVIEW OF THE INTAμSAT'S MASS MEMORY UNIT BASED ON SPACEWIRE[1]

Session: SpaceWire Missions and Applications

Short Paper

J. Ignacio García, J. Antonio Martín, Daniel Meziat, Manuel Prieto

Space Research Group. Dpto. Automática. Universidad de Alcalá. Ctra. Madrid Barcelona Km 33,600. 28871, Alcalá de Henares, Spain

Laura Seoane, Manuel Angulo

INTA. Space Programmes Department., Ctra. de Ajalvir, km. 4. 28850, Torrejón de Ardoz, Spain

E-mail: ignacio.garcia@srg.aut.uah.es, jamartin@srg.aut.uah.es, daniel.meziat@srg.aut.uah.es, manuel.prieto@srg.aut.uah.es, angulom@inta.es, seoanepl@inta.es

ABSTRACT

This paper presents the solutions adopted for the INTAμSAT-1 Mass Memory Unit, which makes use of several high-speed interfaces based on SpaceWire. This MMU handles the information coming from 3 medium resolution Earth observation cameras and other high data rate experiments in parallel by using 13 point to point SpaceWire channels with up to 110 Mbps data rate. The information will be properly formatted, stored and queued for later transmission to ground through additional point to point SpaceWire nodes connected to the spacecraft X-Band and the S-Band modems, and a PTM modulated Laser downlink high speed transmitter. The MMU architecture and preliminary tests over a representative prototype to validate the concept are presented.

1 INTRODUCTION

INTAμSAT-1 will be the first mission of a new small satellite programme with a mass ranging from 80 to 150 kg, and compatible with VEGA, Soyuz-ST and Dnepr launchers. The first INTAμSAT-1 will be a nadir pointing satellite with body fixed solar panels devoted to R&D remote sensing, with a launch tentative date during 2012. This enlarged IμSAT class is a further step after the NANOSAT programme success with a launch onboard Ariane-V-165 in Dec. 2004 (Nanosat-01 is still working in orbit), and the Nanosat-1B, a follow-on UHF store&forward communications mission with new experiments, launched from Baikonur by a DNEPR in July 2009. Inside the INTAμSAT-1 spacecraft two types of data buses are used [1]. The OBDH housekeeping TM/TC data bus, to which all units inside the spacecraft will be connected, will use the CAN standard. The Mass Memory Unit (MMU) implements several SpaceWire high-speed channels for information exchange

[1] This work has been partially supported by the Spanish Ministry of Science and Innovation MICINN under the grant AYA2009-13478-C02-02.

with the payload equipments and the communications subsystem. Its objective is to provide a temporary memory resource in order to store the huge amount of data provided by the cameras until it is downloaded to the ground stations.

2 MMU REQUIREMENTS

2.1 COMMUNICATION INTERFACES AND BANDWIDTH ASSESSMENT

The most defining factor in the design of the MMU is the amount of high-speed channels to be used simultaneously. The satellite features 3 cameras [2] with several CCD outputs, providing many data links even after processing and multiplexing:

- The CINCLUS instrument provides 8 data signals with a bitrate around 18 Mbps. Multiplexing results in 2 links producing 72 Mbps each.
- Each of the two MS-WAC cameras provides 10 data signals with a bitrate around 55 Mbps. Multiplexing results in 10 links producing 110 Mbps each.
- Finally, the PAU experiment requires a single 8Mbps link.

This results in a total of 13 SpaceWire links with a peak data rate of 1,252 Mbps. Data must also be sent to ground, for which the satellite features 3 communications systems: A 100 Mbps laser downlink, a 72 Mbps X-Band modem (including 7/8 Viterbi and 187/204 Reed-Solomon codifications) and an 8 Mbps S-Band modem (including 7/8 Viterbi), resulting in a peak bandwidth of 172 Mbps.

2.2 CAPACITY AND OTHER MEMORY CONSTRAINTS

It's been determined that a total of 80 Gbits of net storage is necessary for the mission. Due to its combination of higher radiation tolerance and reasonably high density, SDRAM technology has been selected for mass storage.

2.3 OTHER CONSIDERATIONS

Apart from the aforementioned constraints, the MMU must feature a CAN bus interface both for command and control, and to provide low-speed storage services for other spacecraft subsystems such as the OBDH. Additionally, the need for achieving low power consumption and the size constraints (160x200mm double-Europe board size), dictate a system with few components and low operating frequency. A single FPGA system with a LEON2 synthesized processor and embedded RAM seems like a good enough target; specifically an Actel RTAX2000 FPGA has been chosen.

3 MMU DESIGN

3.1 PRELIMINARY DESIGN CONSIDERATIONS AND APPROACH

The fact that the MMU needs to address many SpaceWire channels concurrently makes a classical single-CPU data processor approach to the MMU design very difficult. The interrupt rate generated and a data stream well above the 100 MB/s mark would be difficult to handle by a synthesized CPU with a targeted speed of a few tens of Mhz. Direct memory access by the hardware handling the SpaceWire

links is clearly a must, and on-the-fly post-processing and formatting of the stored data seems infeasible. Another matter of concern is that the imposed 80 Gbits storage requirement far surpasses the 4 GB physical addressable memory directly supported by typical 32-bit processors such as LEON2.

A system consisting of several independent SpaceWire controllers with direct access to the mass memory managed by a simple CPU-based controller subsystem seems like a more feasible approach. The controller is unloaded of the burden of moving data back and forth and can be kept as simple as possible. A single-channel 64-bit SDRAM memory seems adequate for the application while working at a reasonable 25Mhz operating frequency (peaking at 200 MB/s data transfer rate).

3.2 DETAILED FUNCTIONAL DESCRIPTION

Given the MMU requisites and design limitations, the only service provided by the MMU is raw storage. Data received from each channel will be directly stored in memory with no further post-processing or formatting, and will be transmitted to ground also in this form. Error checking, data correlation, analysis, etc. will be done in the ground station and the packet format (if any) used by the payloads will be transparent to the MMU. The OBDH is be responsible of instructing the MMU about the memory areas where the data coming from every specific channel is to be stored, or what data is to be sent to the communications subsystem. Thus, the MMU is treated as a raw storage device and it is the OBDH who accounts for free and used space and decides data retransmission or other management decisions.

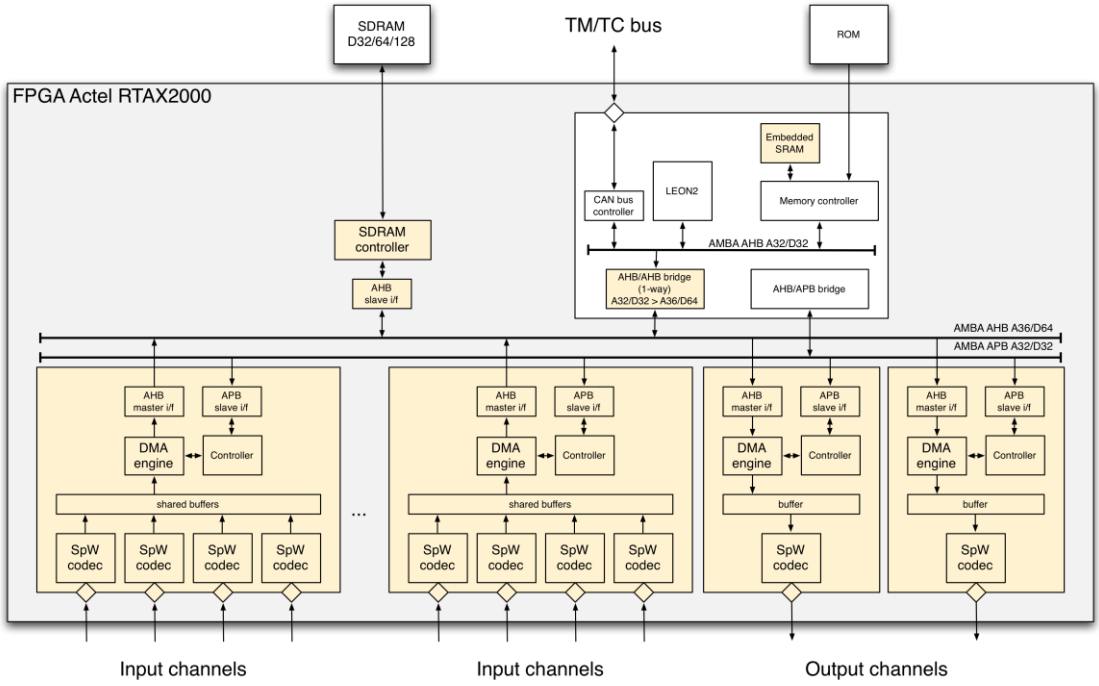


Figure 1: MMU block diagram

3.3 BLOCK DIAGRAM

A SoC design (further explained in the following paragraphs) has been produced, represented in the high-level block diagram on Figure 1. The shaded blocks are in-house developed IP cores, while the light ones are freely available IP cores, slightly

modified versions of these or just discrete components. The design is built around a slightly customized AMBA AHB bus with modifications to the address space range to fit the required addressable mass memory, and wide enough to cope with the required data rate while maintaining a relatively low operating frequency. A 36-bit address, 64-bit data bus has been proposed, potentially leading to a 128-bit data bus implementation. The mass memory chips are connected to an in-house developed SDRAM controller with slave AHB interface, allowing us to make it adaptable to various address and data widths or even different memory technology. This approach allows for easy addition of multiple mass memory banks if necessary, by connecting more slave memory controllers to the AHB bus.

The SpaceWire receiver and transmitter elements are kept as simple as possible, focusing mainly in high-performance and low FPGA footprint. Transfer parameters can be programmed through registers, and once a transfer is activated the IP core can receive/send data from/to the SpaceWire link autonomously, much like a programmable DMA controller. These elements employ an in-house developed SpaceWire IP core codec [3] and have a master AHB interface, which allows them to perform read/write operations independently and provides great flexibility since it allows adding and removing SpaceWire channels in a very straightforward manner. 2 or 4 SpaceWire channels are grouped in the same IP core to keep the number of AHB masters in the bus below the maximum supported.

In order to process TC from the OBDH and produce the required TM, the MMU features a LEON2 SoC with CAN bus interface as controller subsystem, which is able to program the SpaceWire controllers through the AMBA APB bus and perform various other operations. The AHB bus of the LEON2 controller subsystem is independent from the main AHB bus in order not to cripple performance and to override the address and data bus width limitations of the LEON2 CPU. An AHB/AHB unidirectional, programmable bridge has been developed which maps a programmable window of the controller AHB space into the main AHB bus, thus allowing access to the full storage space from the LEON2 controller. Also, since all the IP cores connected to the modified version of the AHB bus are developed in-house, any customization regarding address or data width is possible.

4 IMPLEMENTATION AND PROTOTYPE

To help with IP core testing and to prove the validity of the design, a simpler MMU demonstration prototype has been produced using a Pender GR-CPCI-XC4V Virtex4 FPGA development board with the accessory GR-CPCI-SER2-SPW2 mezzanine, which provides two SpaceWire ports. The prototype features the same design as the final system, but there are only two SpaceWire IP cores, one for input and one for output. Also, the main AMBA AHB bus is only 32 bits wide and supports 32-bit addresses, and all the AMBA AHB masters and slaves are modified accordingly. The two AHB bus approach using the AHB/AHB bridge is used, nonetheless. Finally, the LEON2 controller subsystem does not use an external ROM; the firmware is directly loaded and run from embedded SRAM using the GRMON tool and the on-board SDRAM of the XC4V is used exclusively as mass storage. The firmware running in the MMU prototype has been custom built and does not use any operating system, and despite reduced functionality is basically the same than the firmware that is expected to run in the final system.

In order to emulate the payloads, a Gaisler SpaceWire-RTC development suite has been used. This system features two SpaceWire links which are used to emulate the data stream coming from a payload and a to downlink interface, allowing us to building a simplified scenario which is representative of the final solution. It also features a CAN bus controller which is used to command the emulator, to configure and run the tests. The software running in the RTC uses RTEMS as underlying OS.

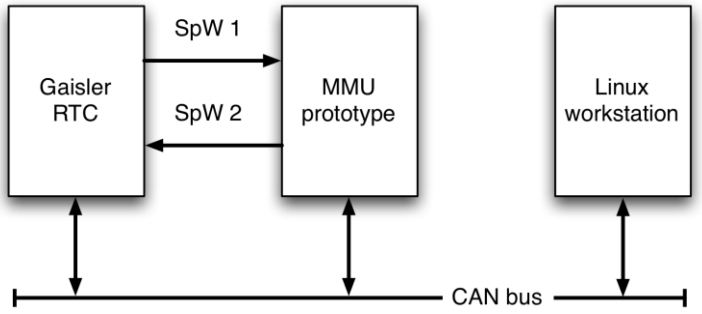


Figure 2: Prototype testing scenario

Finally, a Linux PC workstation with a CAN bus adapter is used to configure and control both the MMU prototype and the payload/downlink emulator in the RTC through a simple to use visual application. The scenario is shown in Figure 2, and has been proved to work as expected with both SpaceWire channels working concurrently and the LEON2 controller handling the configuration and control as instructed.

5 CONCLUSIONS AND FUTURE WORK

A simple MMU design valid for high data rate, high interface count applications has been produced, providing basic raw storage services adequate for certain applications. The design is focused in high bandwidth and simple implementation, sacrificing the provision of more complex storage services in exchange of the ability to use fewer components and lower power. At the same time, the design provides high flexibility to provide an easy upgrade path in case a different channel configuration, more memory banks or higher fault tolerance is needed, and in the presented scenario it requires only the use of a single, medium capacity FPGA. Work is already underway to support DDR and DDR2 memory and to improve the developed IP cores to enhance performance and reduce footprint. Also, more advanced firmware capabilities are being considered to support features such as scheduled data downloading.

6 REFERENCES

1. D. Guzman, M. A. Angulo, L. Seoane, S. Sánchez, M. Prieto and D. Meziat, "Overview of the INTAμSAT's Data Architecture Based on SpaceWire", International SpaceWire Conference, Dundee, Scotland, 2007.

2. M. A. Angulo., L. Seoane, E. Molina, M. Prieto, O. Rodríguez, S. Esteban, J. Palau and E. Cornara , "INTAμSat-1 First Earth Observation Mission", 7th IAA Symposium on Small Satellites for Earth Observation. Berlín, Germany, 2009

3. P. Aguilar, M. Prieto, D. Guzmán, D. García and R. Castillo, "Implementación en SoC reconfigurable de un códec Spacewire basado en el estándar ESA ECSS-E-ST-50-12C", Jornadas de Computación Reconfigurable, Alcalá de H., Spain, 2009

THE PLATO PAYLOAD DATA PROCESSING SYSTEM

Session: SpaceWire Missions and Applications

Short Paper

Philippe Plasson, Gérard Epstein, Loïc Gueguen

LESIA/Observatoire de Meudon, 5 place Jules Janssen 92195 MEUDON, France.

Christophe Cara

CEA Saclay DSM/IRFU/Service d'Astrophysique,

bât. 709 L'Orme des Merisiers, 91191 Gif-sur-Yvette, France.

Bernard Pontet

CNES, DCT/SB/LV 18 avenue E Belin 31401 TOULOUSE Cedex 9 France.

E-mail: philippe.plasson@obspm.fr, gerard.epstein@obspm.fr, loic.gueguen@obspm.fr, christophe.cara@cea.fr, bernard.pontet@cnes.fr

ABSTRACT

The objective of the PLATO mission is to detect and characterize exoplanetary systems. The PLATO payload is made up of 34 very high-precision photometric cameras, each camera having its own CCD focal plane constituted by four 20-million pixel CCDs. The huge amount of raw data produced for each exposure has led the PLATO Payload Consortium to propose a design of the onboard data processing system based on a hierarchical architecture in which the SpaceWire technology is extensively used and plays a key role.

1 INTRODUCTION

1.1 THE PLATO MISSION

PLATO (PLAnetary Transits and Oscillations of stars) is one of the three Cosmic Vision M-class missions which have been approved on February 2010 by the ESA's Science Programme Committee to enter the definition phase.

The objective of PLATO is the detection and characterization of exoplanetary systems of all kinds, including both the planets and their host stars, reaching down to small, terrestrial planets in the habitable zone [1]. The PLATO instrumental concept is based on an ultra-high precision, long (few years), uninterrupted photometric monitoring in the visible of very large samples of bright stars. The resulting high quality light curves will be used on the one hand to detect planetary transits, as well as to measure their characteristics, and on the other hand to provide a seismic analysis of the host stars of the detected planets.

During the assessment phase which has been completed at the end of 2009, studies of the whole mission have been carried out independently by two industrial contractors.

At the same time, an assessment study of the PLATO payload (telescopes, detectors, cameras, on-board data processing system) has been provided by a consortium of research institutes and universities: the PLATO Payload Consortium (PPLC). The design of the PLATO payload data processing system presented in this paper is the result of the PPLC studies and will serve as a baseline for the further studies of the definition phase.

1.2 INSTRUMENTAL CONCEPT

The instrumental concept proposed by the PPLC is based on a multi-camera approach, involving a set of 32 normal cameras monitoring stars fainter than mV=8, plus two fast cameras observing extremely bright stars with magnitudes lower than 8. The 32 normal cameras are arranged in four sub-groups of 8 cameras. All 8 cameras of each sub-group have exactly the same field of view, and the lines of sight of the four sub-groups are offset by half the size of the field of view. This particular configuration allows surveying a very large square field at each pointing, with various parts of the field monitored by 32, 16 or 8 normal cameras. This strategy optimizes both the number of targets observed at a given noise level and their brightness.

Each normal camera is equipped with its own CCD focal plane array (FPA) constituted by four 20-million pixel CCDs working in full frame mode. The fast cameras are equipped of four 10-million pixel CCDs working in frame transfer mode. With 32 normal cameras working at the cadence of 25 seconds and 2 fast cameras working at the cadence of 2.5 seconds, the amount of raw data produced by the PLATO payload at the output of its 136 CCD detectors is close to 189 Terabits per day. This volume must be compared to the 109 Gigabits which can be actually downloaded each day to the ground. It is clearly not possible to transmit the whole amount of raw data. The role of the on-board treatments will be to reduce by a factor of more than 1700 the flow rate by downlinking light curves, centroid curves and imagettes at the cadence required by the science.

This huge amount of data to process has led the PPLC to propose a design of the onboard data processing system based on a hierarchical architecture in which the SpaceWire technology is extensively used and plays a key role at each stage of the architecture.

2 OVERVIEW OF THE PLATO PAYLOAD DATA PROCESSING ARCHITECTURE

The PLATO payload data processing system is made up of 34 Data Processing Units (DPUs) responsible for reducing the data flow by computing light and centroid curves. The 34 DPUs are connected to two central Instrument Control Units (ICUs) through a SpaceWire network built around ten SpaceWire routers to which ten other routers are added for the redundancy management. The ICUs are working in cold redundancy and are connected to the spacecraft Service Module (SVM) through SpaceWire links.

There is one DPU per camera performing the basic photometric tasks and delivering a set of light curves, centroid curves and imagettes to the active ICU, which stacks and compresses the data, then transmits them to the spacecraft SVM for downlink. Data from all individual cameras are transmitted to the ground, where final instrumental corrections, such as jitter correction, are performed. The DPUs of the fast cameras

will also deliver a periodic pointing error signal to the spacecraft AOCS (Attitude and Orbit Control System). Several photometry algorithms (aperture photometry, weighted mask photometry, Line Spread Function fitting) are planned to run on board, each star being processed by one of them, depending on its brightness and level of confusion.

For FMEA (Failure Mode and Effects Analysis) concerns and in order to optimize the resources (mass, volume, harness), the DPUs of the 32 normal cameras are distributed in 8 groups of 4 DPUs. Each group of 4 DPUs is gathered in a box called a Main Electronic Unit (MEU). A set of 2 MEUs corresponds to a set of 8 cameras sharing the same field of view. A MEU gathers in the same box four DPU boards, 1 power supply (DC/DC converter) and 2 SpaceWire routers to merge the data from the DPUs toward the ICU (one main and one redundant).

All the command / data exchanges between the payload units and with the spacecraft SVM are ensured thanks SpaceWire links except the AOCS link between Fast DPU and the spacecraft SVM. The figure 1 shows the architecture of the PLATO data processing system and its SpaceWire network.

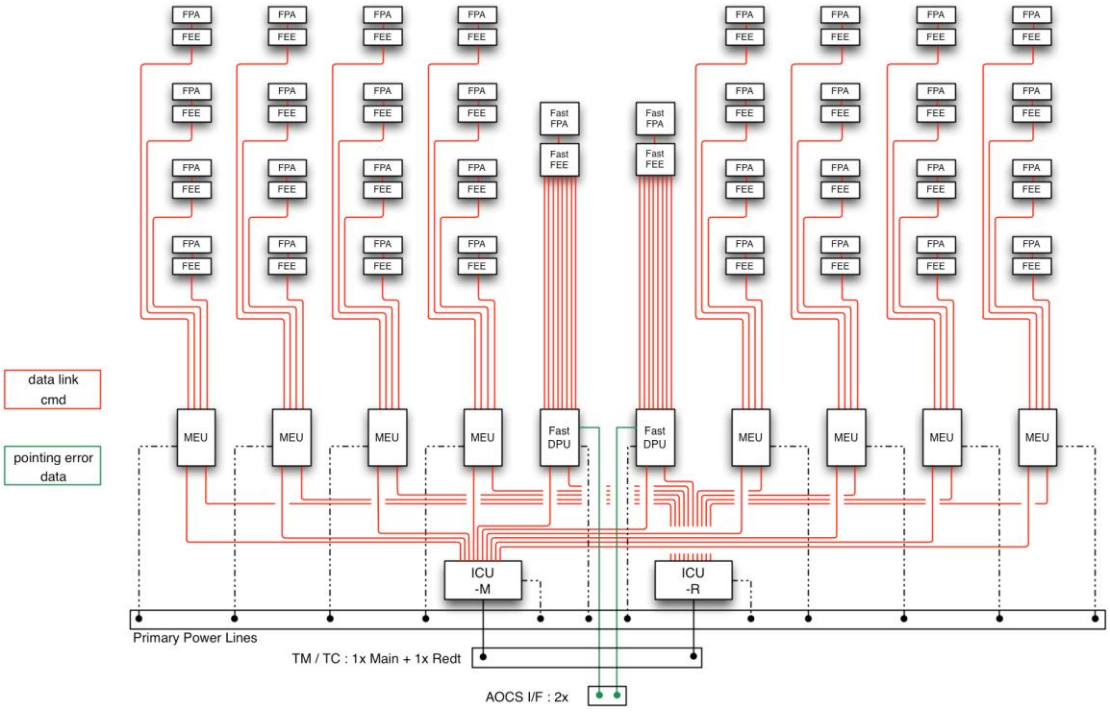


Figure1 – PLATO On-board Data Processing Architecture

3 THE DATA PROCESSING UNITS (DPUS)

Each normal camera has its own front-end electronics (FEE) driving the four CCDs and transmitting the pixel data to its DPU through a SpaceWire link. At the FEE level, no pre-processing such as windowing is done. The full images corresponding to each CCD are transmitted entirely. The raw data acquired from the ADC are just serialized and sent to each normal DPU through one point-to-point SpaceWire channel. The communication between the FEE and the DPU uses a point-to-point SpaceWire channel without any router in order to guarantee the real-time constraints on this interface. Over a period of 25 seconds corresponding to the CCD readout period, each

normal DPU receives 4 CDD full-frame images (one 330-Mb image / 6.25 seconds). Each CCD full-frame image is entirely stored in SDRAM memory before processing. The RMAP protocol [3] is used to transfer at a very high rate close to 160 Mbps the full frame images directly to the DPU memory without any cost for the DPU software. The CPU occupation rate needed to acquire and to store the full-frame images shall be negligible: this is a strong requirement. The CPU resources shall be reserved for the photometric treatments of the 120.000 stars distributed on the four CCDs. The DPU hardware module in charge of the image reception relies on the DMA (Direct Memory Access) technology and provides a hardware implementation of the RMAP protocol. Using DMA and RMAP technologies, it could be possible to have an image transfer fully transparent for the CPU: the RMAP protocol is particularly well suited for this kind of application. The core of the normal DPUs will be a LEON-FT processor chip ideally integrating SpaceWire interfaces with RMAP and DMA support.

Unlike the normal DPUs, the fast DPUs receive the pixel data from the FEE of their respective camera through 8 specific LVDS high speed serial links (one link per CCD output): using 8 SpaceWire links was considered too costly in term of resources and merging the 8 outputs to one channel leads to a flow rates greater than 600 Mbps which is not compatible with the SpaceWire performances.

4 THE INSTRUMENT CONTROL UNITS (ICUS)

Both ICUs work in cold redundancy. They play the role of the payload conductor: they schedule the DPU tasks by the way of commands, manage the mode changes and regulate the data flow. They are connected to the eight MEUs and to the two Fast DPUs through SpaceWire links. The ICUs use the SpaceWire network to propagate a synchronization signal to all the DPUs thanks to the SpaceWire time-code functionality [2]. They are also responsible for formatting and transmitting to the spacecraft SVM the scientific telemetry packets and the housekeeping telemetry packets. Each ICU is connected to the SVM with two SpaceWire links (one main and one redundant) for telecommand reception and telemetry transmission.

In the current design, two SpaceWire routers are foreseen in each ICU. An alternative solution, which consists to put the SpaceWire router function outside the ICU box, with the aim of reducing the number of links directly connected to the ICU and also to improve the redundancy scheme, will be studied in the definition phase of the mission.

In the global architecture, the ICUs are also in charge of a second level of data reduction. All the flux and centroids sent by the normal DPUs and the fast DPUs at the cadence respectively of 25 seconds and 2.5 seconds are cross-checked, stacked and temporally averaged in order to produce the final telemetry at the cadence required by the science (50 seconds and 600 seconds). Lastly, before being transmitted to the SVM, the data are compressed using a lossless compression algorithm.

In order to reduce the complexity of the DPUs, the in-flight maintenance of the DPU application software is delegated to the ICUs. The DPUs don’t have locally an EEPROM and don’t manage the maintenance of the application software. The ICUs are responsible for storing in their EEPROM the DPU application software and managing its maintenance (changing the content of the EEPROM, adding a new

version of the application software in EEPROM). After a switch-on or a DPU reset, during the boot process, the DPU boot loader (stored in the DPU PROM) loads the Application Software code and data over the SpaceWire link from the ICU.

5 CONCLUSION

The design of the PLATO payload data processing system is well advanced. However, several issues must still be consolidated until the end of the definition phase and the final selection of the Cosmic Vision M-class missions by ESA which is expected for June 2011. Firstly, the technical budgets of each subsystem (DPUs, ICUs) concerning the CPU resources, the power consumption, the surface and the mass will be refined. The design of the different units will be then detailed. Finally, the critical functions and interfaces which have been identified like the high-throughput interface between the FEEs and the DPUs of the normal cameras will be prototyped.

The future studies will be pursued always keeping in mind the two main principles that have governed the preliminary studies. The first principle is to have standard interfaces between the different units of the payload in order to minimize the specific developments at the instrument level and at the test equipment level, to simplify the tests and finally to reduce the global cost. As we have shown in this paper, SpaceWire technology is the ideal response to this requirement. The second principle is to get the most compact design and the less power expensive solution possible at the subsystem level. There are 34 DPUs: 1 watt saved is 34 watts saved. The goal is then to minimize as much as possible the complexity of the DPUs: if a function can be delegated to the ICU, then it shall be delegated to the ICU.

The PLATO definition phase will be also an opportunity to address the complex question of the data processing system AIT (Assembly, Integration and Tests) and to propose a global approach for conducting these activities.

6 REFERENCES

1. European Space Agency, “Plato, Next-generation planet finder. Assessment Study Report”, ESA/SRE(2009)4, December 2009.

2. ECSS, “SpaceWire: Links, nodes, routers and networks”, ECSS-E50-12A, January 2003

3. ECSS, “SpaceWire protocols”, ECSS-E-ST-50-11C Draft 1.3, July 2008, pp.20-109

SPACEWIRE APPLICATION FOR THE X-RAY CCD CAMERA ONBOARD THE ASTRO-H SATELLITE-THE BBM DEVELOPMENT AND THE EM DESIGN

Session: SpaceWire Missions and Applications

Short Paper

Naohisa Anabuki, Kiyoshi Hayashida, Rui Sakaguchi, Masashi Kimura, Hiroaki Takahashi, Hiroshi Nakajima, Hiroshi Tsunemi, Masaharu Nomachi

Osaka University, 1-1 Machikaneyama, Toyonaka, Osaka 560-0043, JAPAN

Masanobu Ozaki, Takahisa Fujinaga, Keiko Matsuta, Aya Bamba, Tadayasu Dotani, Hirokazu Odaka, Tadayuki Takahashi, Motohide Kokubun, Takeshi Takashima

Institute of Space And Astronautical Science/JAXA, 3-1-1 Yoshinodai, Sagamihara, *Kanagawa 252-5210, JAPAN*

Takayuki Yuasa

The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-0033, JAPAN

and the SXI team

E-mail: anabuki@ess.sci.osaka-u.ac.jp, hayashida@ess.sci.osaka-u.ac.jp, sakaguchi@ess.sci.osaka-u.ac.jp, mkimura@ess.sci.osaka-u.ac.jp, h-takahashi@ess.sci.osaka-u.ac.jp, nakajima@ess.sci.osaka-u.ac.jp, tsunemi@ess.sci.osaka-u.ac.jp, nomachi@lns.sci.osaka-u.ac.jp, ozaki@astro.isas.jaxa.jp, fujinaga@astro.isas.jaxa.jp, matsuta@astro.isas.jaxa.jp, bamba@astro.isas.jaxa.jp, dotani@astro.isas.jaxa.jp, odaka@astro.isas.jaxa.jp, takahashi@astro.isas.jaxa.jp, kokubun@astro.isas.jaxa.jp, takashima@astro.isas.jaxa.jp, yuasa@juno.phys.s.u-tokyo.ac.jp

ABSTRACT

We are developing Soft X-ray Imager (SXI), an X-ray charge-coupled device (CCD) camera system on board the ASTRO-H mission. It has a SpaceWire (SpW) interface and will be installed in the satellite's SpW network. In this paper, we report the results of the breadboard model development and present the engineering model design and architecture under consideration.

1 INTRODUCTION

ASTRO-H, Japan's 6th X-ray astronomical satellite, is planned for launch in FY2013. It employs SpaceWire (SpW)-based information network system. And the onboard equipment communicates through the SpW network. The Soft X-ray Imager (SXI), an X-ray charge-coupled device (CCD) camera system in conjunction with the X-ray telescope, is one of the observation instruments. Therefore, the SXI also have to be designed on the assumption that it will be installed in the satellite's SpW network. Since the SpW-based X-ray CCD camera system is a new item we develop, the breadboard model (BBM) has been developed first, and the engineering model (EM) design is currently designed.

2 SXI DIGITAL ELECTRONICS

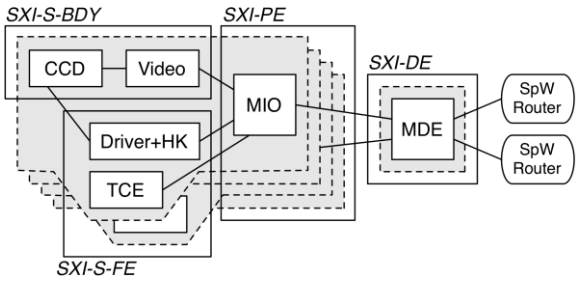


Figure 1: Schematic block diagram of SXI electronics, which consists of four identical circuits and a CPU unit.

Figure 1 shows the component diagram of the SXI electronics. Four X-ray CCD chips are in the focal plane, and each CCD chip has the identical driving and readout circuits. The digital electronics is divided into two processing units, an FPGA-based unit (SXI-PE; Pixel-processing Electronics) and a CPU-based unit (SXI-DE; Digital Electronics). Both of them have SpW I/Fs and communicate with each other via SpW network. By RMAP commands from the SXI-DE, the SXI-PE controls peripheral devices, processes CCD pixel data, and collects HK information. The SXI-DE also handles advanced data processing such as an X-ray event extraction.

2.1 THE HARDWARE

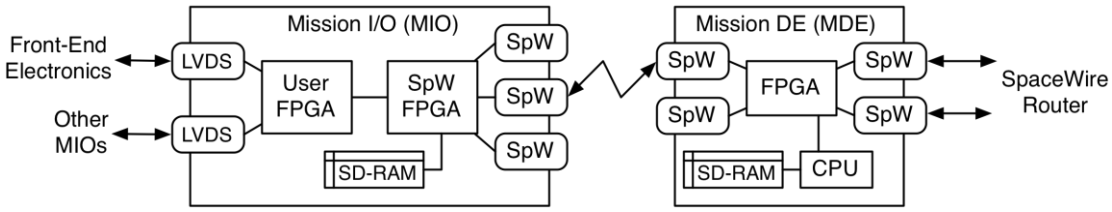


Figure 2: Schematic block diagram of the MIO board and the MDE board.

The hardware devices for the SXI-PE and the SXI-DE are a Mission I/O (MIO) board and a Mission DE (MDE) board, respectively. "Mission" means that these devices have been developed as BUS equipment in the ASTRO-H mission. That enables us to reduce the costs, speed up the development time, and improve the reliability. Figure 2 shows a schematic block diagram of the MIO and MDE boards. The MIO consists of "User FPGA" in which implemented user logic to serve the purpose of each instrument and "SpW FPGA" that has SpW interfaces. Instrument-specific programs and common SpW interfaces are also installed in the MDE.

2.2 THE BBM DEVELOPMENT

We have manufactured the breadboard model (BBM) of the SpW-based CCD camera system, whose DAQ structure is described in Fujinaga et al. 2010 (this conference) in detail. While the BBM hardware is slightly different from those of we show in Figure 2, the basic configuration and functional allocation are similar. We have developed minimum required functions to obtain CCD raw image and implemented them in the BBM system. The BBM works well in the lab system.

3 THE ENGINEERING MODEL DESIGN

We have been working to design the SXI engineering model, which is a SpW/RMAP -based architecture of CCD driving and data handling using the SXI-PE and the SXI-DE. The block diagram of the SXI-PE processing unit is shown in Figure 4.

3.1 ARCHITECTURE

Here, we represent functions closely associated with SpaceWire/RMAP communication, while there are a lot of SXI-specific modules implemented in the User FPGA of the SXI-PE. In the current EM design of the MIO board, a relatively large-capacity memory (64 MByte SD-RAM) is connected to only the SpW FPGA. However, the CCD data processed in the User FPGA need to be transferred to the memory. Therefore, to utilize an intervening (board-local) bus between User FPGA and SpW FPGA efficiently, we introduced buffering modules, "Bus I/F agents", there. The Bus I/F agents have double buffers and regulate the data flow from the upstream to the downstream. Read and write from and to the double buffers are controlled by "Address manager". When the one-sided buffer of the Bus I/F agent is filled with the processed data, the address manager switches the buffers' I/O, sets the destination address, and triggers the date transfer to the target SD-RAM in accordance with established priorities of the stored data. The SXI-DE, on the other hand, access stored data in the SD-RAM by multiple RMAP read. The RMAP read is kept waiting by the SpW FPGA of the MIO while the target buffer memory is updating. The interface is "Exposure information" in the SD-RAM. After all the data transfer from the User FPGA to the SD-RAM is complete, the "Address Manager" unlocks the exposure information. HK data are collected during periodic intervals in the spare time.

3.2 SPACEWIRE NETWORK TOPOLOGY

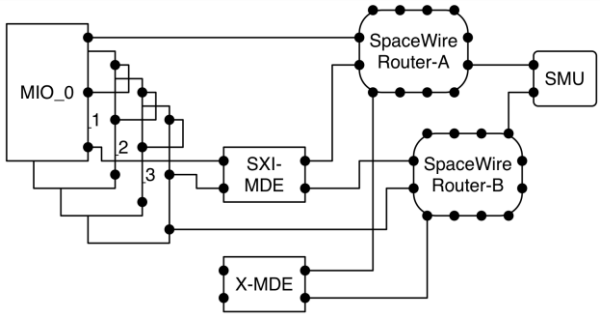


Figure 3: SpaceWire network topology for the ASTRO-H SXI.

We are also considering the SpW network topology for SXI to avoid single point of failure (SPOF) for the SXI-DE since SXI has only one CPU unit. Figure 3 shows the current design of he SpW topology to realize redundant configuration for the SXI system. Connecting at least two MIOs to SpaceWire routers (A and B), the route to use the common backup DE (X-MDE) via the SpW router is secured.

4 REFERENCES

1. Ozaki, M. et al. "NeXT SXI data processing system", Proceedings of SPIE Space Telescopes and Instrumentation 2008: Ultraviolet to Gamma Ray

2. Ozaki, M. et al., "SpaceWire driven architecture for the ASTRO-H satellite", this conference, 2010

3. Fujinaga, T. et al. "Development of SpaceWire based data acquisition system for the X-ray CCD camera on board ASTRO-H", this conference, 2010.

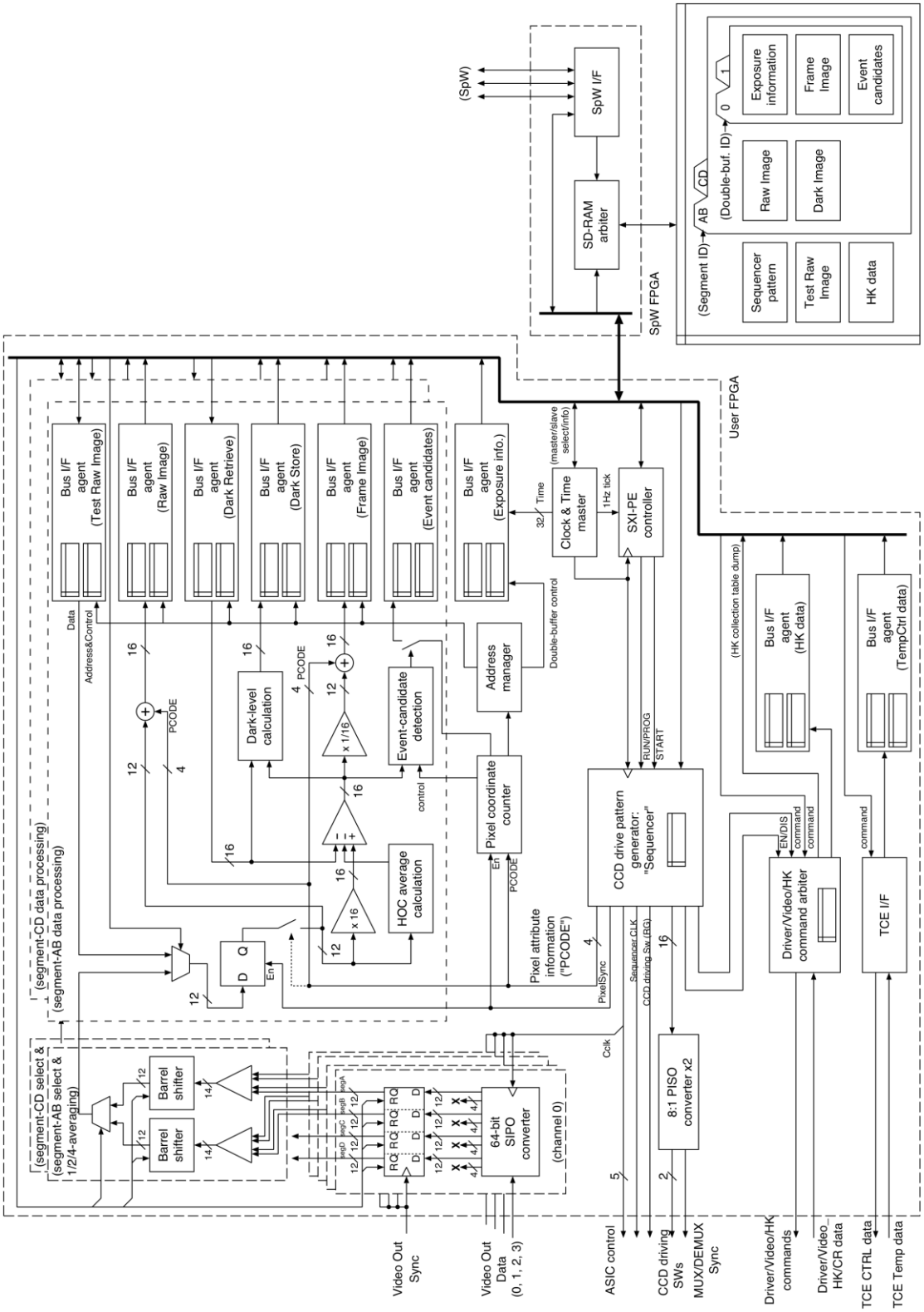


Figure 4: The block diagram of SXI-PE processing unit, which is for one CCD chip. Modules with the same function are displayed as layers.

Papers Indexed by Session

Tuesday 22 June

Standardisation (75 mins)

Test and Verification 1 (45 mins)

Test and Verification 2 (90 mins)

Wednesday 23 June

Components 1 (100 minutes)

Components 2 (110 minutes)

Poster Presentations

Components 3 (45 minutes)

Onboard Equipment and Software (65 minutes)

Thursday 24th June

Networks and Protocols 1 (105 minutes)

Networks and Protocols 2 (105 minutes)

Missions and Applications 1 (80 minutes)

Missions and Applications 2 (100 minutes)

*Full paper not included in proceedings

(L) Long paper - 20 minutes
(S) Short paper - 15 minutes
(P) Poster paper

Papers Indexed by Author

First Author's Surname A-J

First Author's Surname K-R

First Author's Surname S-Z

Exhibitors

Since 1990, **Shimafuji Electric** has been developing microcomputer boards including transmission, graphics and other complex peripheral functions and also producing small amount of products for some OEMs. We have more chances to develop evaluation boards for various RISCs and intelligent peripheral functions devices and T-Engine boards/T-Engine appliance products these days.
We developed the Space Wire compliant cubic computer, Space Cube with Japan Aerospace Exploration Agency, 5 years ago, and we have some Space Wire function boards, like Sampling ADC, Digital I/O, and ETC.
We also had developed the world smallest one board computer (50mm x 50mm) that power consumption is 1.5W (or smaller).

STAR-Dundee specialises in supporting users and developers of SpaceWire technology by providing:

- Development equipment: Our products cover everything needed to design, develop, integrate and test SpaceWire sub-systems.
- Chips and IP Cores: Enabling our customers to develop their own flight sub-systems and providing custom IP cores to fulfil specific customer needs
- Bespoke Design Services: Equipment and design of electronic circuit boards for custom requirements.
- SpaceWire Training: Onsite expert tuition direct from our experienced engineers, tailored to suit the customer

The STAR-Dundee team has leading expertise in all areas of SpaceWire technology. Our commitment is to help our customers quickly and efficiently get up to speed with SpaceWire technology and support the full development life cycle.

Website: www.star-dundee.com
Email: enquiries@star-dundee.com

www.aeroflex.com/HiRel

Aeroflex Colorado Springs is a supplier of integrated circuits and custom circuit card assemblies. We supply a broad range of standard products for HiRel applications including a LEON 3FT microprocessor, logic, FPGAs, memories, serial communication interfaces for MIL-STD-1553, 1773, Clocks, an LVDS family of products and our SpaceWire products -Transceivers, Protocol IP, Routers. Our RadHard-by-Design Digital and Mixed-Signal ASICs handle design complexities up to 3,000,000 usable gates.

Aeroflex Gaisler is a provider of SoC solutions and IP-cores for exceptionally competitive markets such as Aerospace, Military and Commercial applications. The Aeroflex Gaisler's IP-cores consist of user-customizable 32-bit SPARC V8 processor and floating-point-unit cores, SpaceWire cores, peripheral IP-cores and associated software and development tools. Aeroflex Gaisler solutions help companies develop application-specific SoCs that are highly competitive for customer specific applications. Gaisler Research's personnel have extended design experience, and have been involved in establishing standards for ASIC and FPGA development.

www.4links.co.uk

* **4Links** test equipment is the de-facto SpaceWire reference, with unparalleled maturity in our design and an unparalleled record of finding errors, and providing the information to correct them;
* The family includes bridges, diagnostic interfaces, routing switches, and monitors, a time interface (IRIG-B) plus an RMAP responder to give hardware response times - all controlled from a single (possibly remote) PC;
* Products interface to Ethernet and Internet, able to be interfaced with virtually any computer, any OS, any where;
* All products are available with connectors for synchronization and triggers, so that multiple test units can be synchronized and recordings time tagged consistently between different computers and discs.
* All products are reconfigurable, using a plug-in memory card, to provide a new or enhanced version or a completely different function to allow hardware re-use for lower cost of ownership.

4Links Limited, Suite EU2, Bletchley Park, Milton Keynes, MK3 6EB, United Kingdom
+44 1908 642001
info@4Links.co.uk
www.4Links.co.uk

FRI Submicron was created in 1989, located in Zelenograd (Moscow) and more than 400 employees are working there. FRI Submicron designs and produces components, modules, devices of the computational embedded equipment for different real-time objects (spacecrafts and aircrafts including). It produces devices for about 20 space objects per a year and all processes are certificated in accordance with ISO 9001-2001. Submicron solves the following tasks: development of equipment for space control systems, development of equipment for aviation control and detection systems, development of equipment of a satellite communication and navigation, development of reception and processing systems of the radar-tracking and hydroaudio information, development of the tool, bench and built software.

Yuzhnaya promyshlennaya zona, proezd 4806, building 4, stroenie 2
Zelenograd, Moscow, 124460, Russian Federation

http://submicron.ru
Phone 1 +7 499 731 9651
Phone 2 +7 903 724 2649
Fax +7 499 731 2753
E-mail submicron@se.zgrad.ru

“ELVEES" RnD Center is a leading Russian ASIC design house, number one in the Multicore digital signal processors and «systems on a chip (SOC)» with SpaceWire links: routers, adapters, controllers – the largest chipset in Russia for the space and telecommunications, navigation and embedded systems. ELVEES has its own innovative MULTICORE IC design platform which includes a great 0.25 - 0.65u silicon proven IP - cores library (SpaceWire IP – cores also), based on the commercial 0.25-u CMOS RadHard/temperature stability libraries suitable for space. ELVEES provides for its chips the Tools and Application Software for image compression, adaptive signal processing, optical and radar monitoring, artificial vision, telecommunications and navigation applications.

Contact: Jean-François **Cazaux**
Tel.: +33 (0)5 61 41 77 03
Email: cazaux@skylab-corporate.com
Web: **www.skylab-corporate.com**

SKYLAB Industries is a French company of 15 employees located in Toulouse, the Aerospace Valley country. Our team helps our customers to succeed and grow in **aeronautics and space projects**. We give them a competitive advantage thanks to our know-how and experience focused on **embedded systems** development. Our main activities are:

- Industrial Engineering Department (IPSE) **designs and realizes mains supply systems on VIP aircrafts**;
- Projects Expertise Department (IQAS) **offers services in Quality and Product Assurance management**;
- Software and Electronics Department (EASE) **develops products and solutions for embedded equipments tests**.

Skylab Industries acquires expertise and invests in research and development to fuel **products innovation**. For instance, our engineers worked for almost one year, in partnership with CEA/IRFU for the IP core, on a new products line for test equipments using **SpaceWire** protocol**: compatibleCable^{4SpW}, PCI^{4SpW} board, PCI Express^{4SpW} board, smartCable^{4SpW} and TraffiController^{4SpW}**.

Actel Corporation (USA, California) is a global leader in radiation-tolerant FPGA development and production for aerospace applications. High density gate arrays both reprogrammable and one-time programmable are best possible solution to develop customized onboard electronics using "system-on-chip" technology. Actel technical support and sales center in Russia (Actel.ru, St.Petersburg) is involved in custom design of fault-tolerant flight computers and SpaceWire technology practical application.

Contacts: Address: 212, Moskovsky pr., Saint-Petersburg, Russia, 196066

Contact telephone number: +7 812 740-62-09
Fax number: +7 812 740-62-09
Web-site: www.actel.ru
e-mail: sales@actel.ru

NEC Corporation is one of the world's leading providers of Internet, broadband network and enterprise business solutions dedicated to meeting the specialized needs of a diversified global base of customers. NEC delivers tailored solutions in the key fields of computer, networking and electron devices, by integrating its technical strengths in IT and Networks, and by providing advanced semiconductor solutions through NEC Electronics Corporation. The NEC Group employs more than 140,000 people worldwide. For additional information, please visit the NEC Web site at: www.nec.com.

Contacts:	Kosuke Yamauchi, +81-3-3798-6511	k-yamauchi@ce.jp.nec.com
	Joseph Jasper, +81-3-3798-6511	j-jasper@ax.jp.nec.com