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.

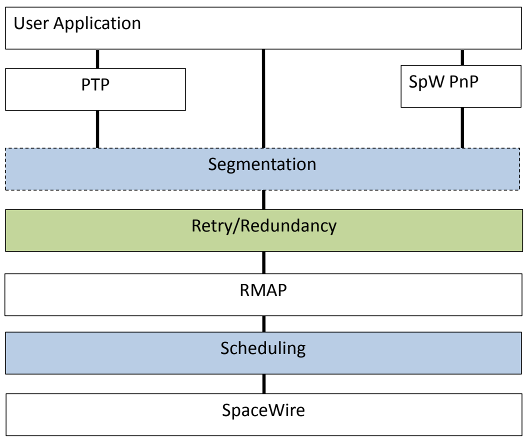

Figure 2: SpaceWire protocol stack using SpaceWire-D

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 .