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.

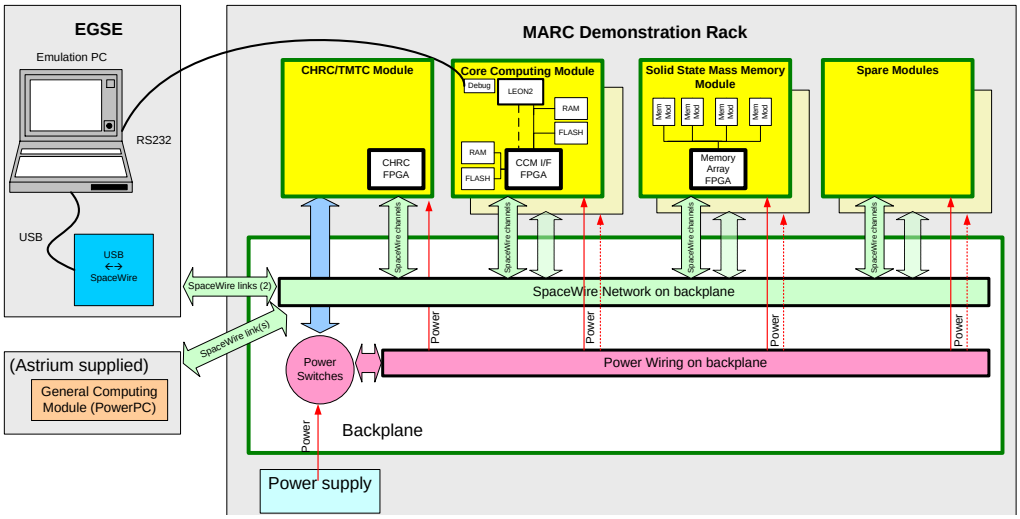

Figure 1. MARC Demonstrator Architecture

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.

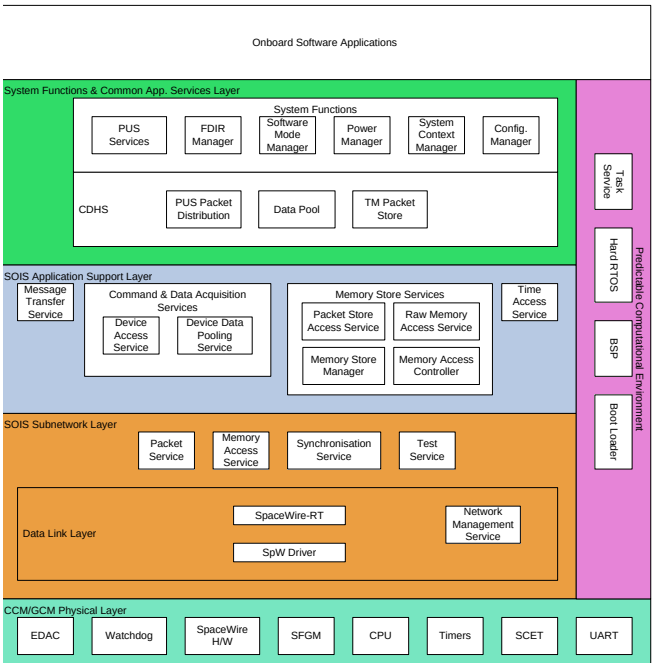

Figure 2. GenFAS Software Architecture

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.

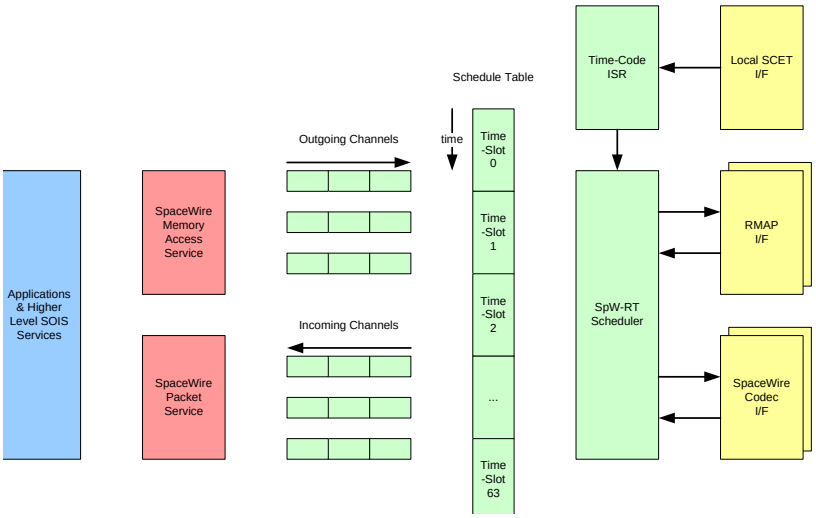

Figure 3. SpaceWire-RT Implementation Block Diagram

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.