MODULAR ARCHITECTURE FOR ROBUST COMPUTING (MARC)

Session: Onboard equipment and software

Short Paper

Alan Senior & Philip Ireland

Systems Engineering & Assessment Ltd

Stuart D. Fowell & Roger Ward

SciSys UK Ltd

Dr. Omar Emam & Dr. Ben Greene

EADS Astrium Ltd

E-mail: alan.senior@sea.co.uk, philip.ireland@sea.co.uk, stuart.fowell@scisys.co.uk, roger.ward@scisys.co.uk, omar.emam@astrium.eads.net, benjamin.greene@astrium.eads.net

ABSTRACT

This paper describes the Modular Architecture for Robust Computing (MARC) concept, a modular processing system that is interconnected by a SpaceWire network. Additional processing, memory or IO modules may be added to improve system availability and/or performance. Future tests on the demonstration system will verify the FDIR system analysis and performance predictions.

The hardware architecture is closely coupled to the software aims of Generic Fault-tolerant Software Architecture using SOIS (GenFAS) software framework, which allows software builds to be allocated to available processor modules, and re-allocated in the event of a failed processor module, enabling a reduced amount of redundant processing modules through an n+m rather than 2n provision.

1 INTRODUCTION

The aim of the Modular Architecture for Robust Computing (MARC) project is to demonstrate the essential features of a heterogeneous, fault tolerant, high availability distributed avionics system based on a SpaceWire network, to a point where it is considered to be a Product that is "one-step-from-flight".

The derived MARC architecture must provide a scalable solution that can meet the demanding needs of future missions. For example, the SpaceWire network architecture must be scalable to include new functions and to provide duplicate paths to achieve the level of redundancy needed for a particular mission.

An important aspect of the demonstrator hardware is that the key components are space qualifiable parts; permitting the design to be upgraded to a fully space qualified system with minimal changes. Similarly the existing GenFAS OBSW software framework will be upgraded to a Product in accordance with space qualified software

standards (ECSS-E-40) and implementing the Spacecraft Onboard Interface Services (SOIS) communication standards as mapped onto SpaceWire.

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. To that end, an additional key objective is to demonstrate the applicability of the MARC architecture to the ExoMars Rover CDMS and PDHS and show that it can provide a viable solution option.

The FDIR handling features and associated test and validation shall be focused on the robustness of the SpaceWire network and its ability to reliably provide a medium for distributing both data and commands. The integrity of the commanding infrastructure is assessed in terms of guaranteed delivery and response while still complying with the stringent requirements imposed on command delivery, detection and execution latency by time critical embedded systems used for space applications. It is anticipated that the outcome of the MARC FDIR and system performance testing will show that the SpaceWire can be used alone to communicate critical data and commands and in principle replace the traditional methods of command and control distribution such as MIL-STD-1553B.

2 HARDWARE ARCHITECTURE

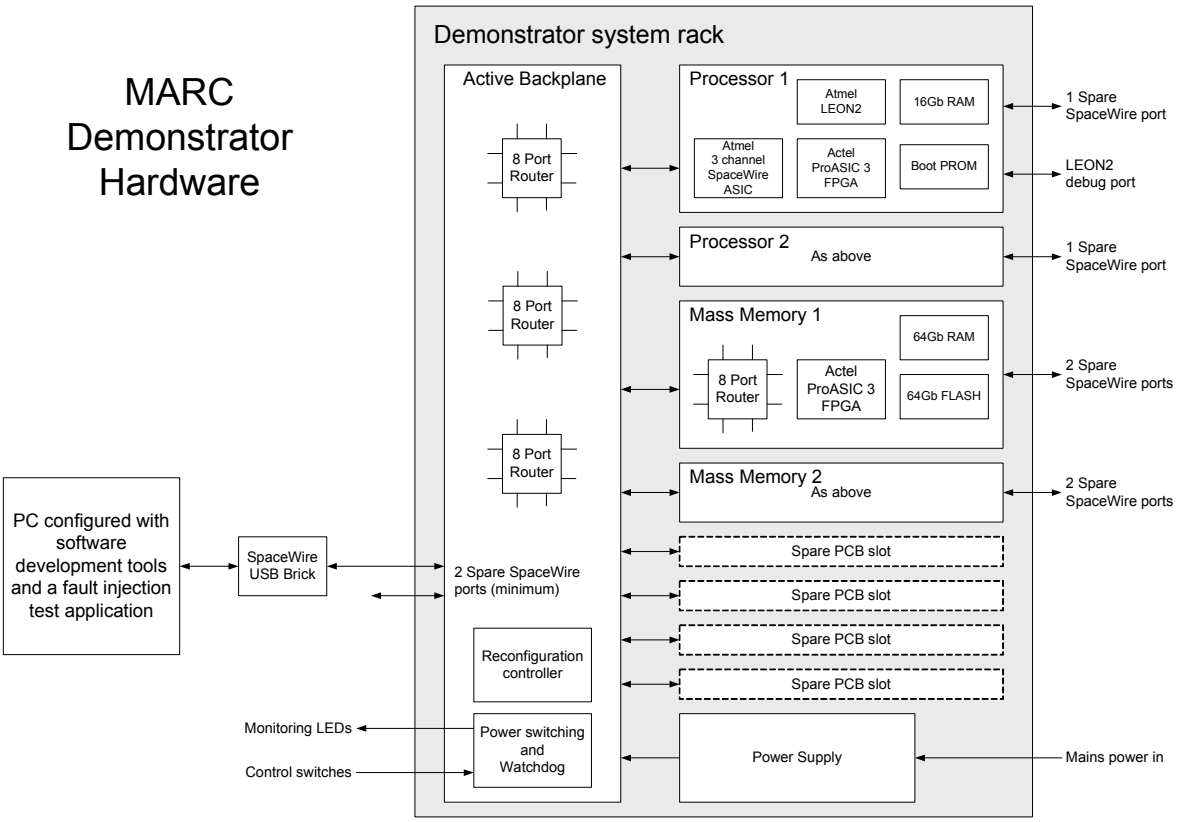

Figure 1: MARC Demonstrator Hardware Block Diagram

Figure 1 is a block diagram of the proposed MARC demonstrator system; it comprises the following principal functions:

- MARC rack containing the modules, backplane and power supply

- EGSE PC loaded with support tools software development and control of the hardware fault injection facilities

- USB brick to interface between the EGSE PC and the MARC rack

The developed modules for the demonstration rack are anticipated to be be:

- A Core Computing Module based on the LEON2 processor

- A 128Gb Mass Memory module incorporating an 8-port SpaceWire Router

- An active backplane incorporating 8-port SpaceWire Routers, power switching and a reconfiguration controller

Spare SpaceWire ports and spare module slots will be provided to permit expansion of the demonstrator at a future date. The Active Backplane incorporates three 8-port SpaceWire Routers. Of the 24 available SpaceWire ports on the backplane, 6 will be used to connect the routers together, leaving 16 available for the 8 backplane PCB slots with the remaining 2 for the spare external SpaceWire ports.

3 FAULT TOLERANT SOFTWARE

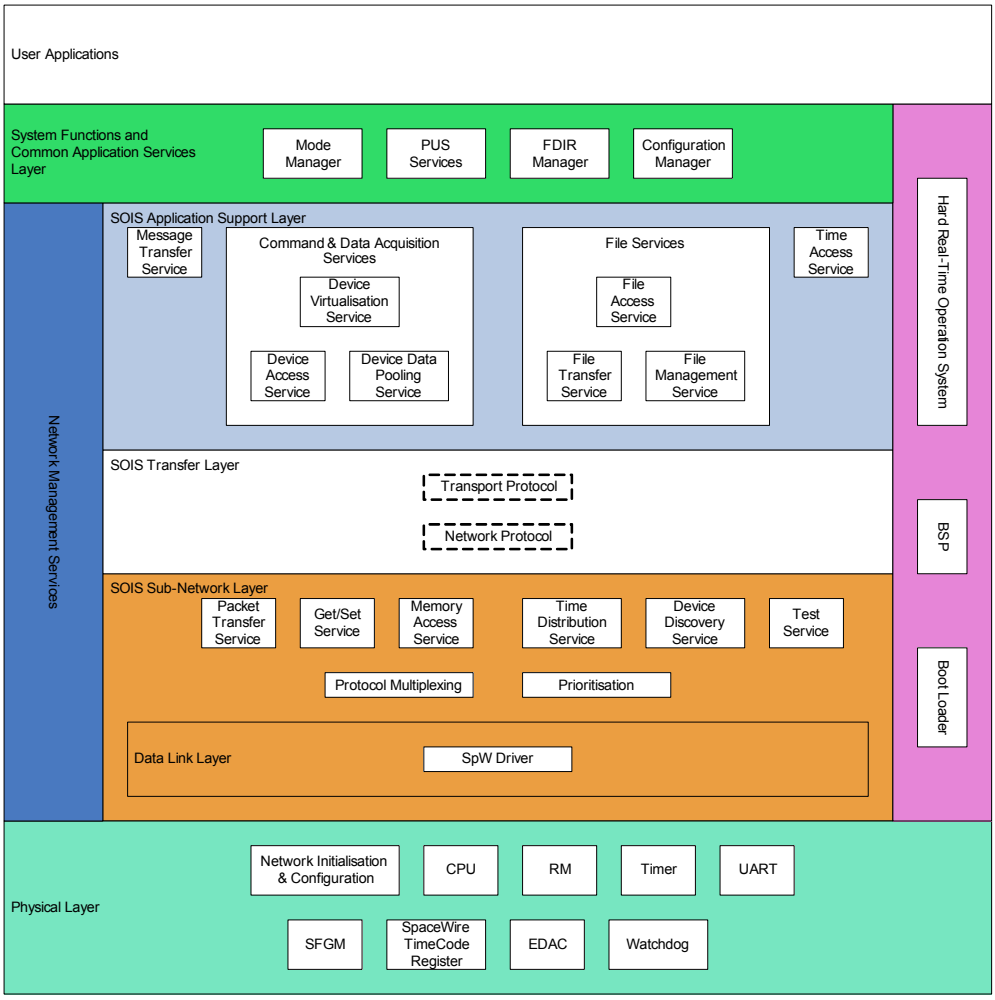

Figure 2: GenFAS Architecture

The Generic Fault-tolerant Architecture using SOIS (GenFAS) [2] is a distributed software framework for Onboard Software (OBSW) applications, implemented to ECSS-E-40 standards [3]. It is deployed upon the processing modules of the MARC demonstrator, has an architecture illustrated in figure 2 and provides the following features:

- **decentralised, distributed OBSW** – PUS Services [4] and a Data Pool are provided to OBSW applications that may be located upon any processing module in the MARC system. For example, TC packets are routed to the destination application based on APIDs and carried using the SOIS services [5] mapped onto SpaceWire protocols [6].

- **decentralised, distributed access to instruments and transducers** – all instruments and transducers are attached either directly to the SpaceWire network or via IO modules and are accessed by applications running on any processing module using the SOIS service libraries mapped onto SpaceWire protocols. Because of this an application running on any processing module has access to all instruments and transducers.

- **advanced mass memory architecture** – GenFAS implements the User libraries and System Controller of the MAGUS architecture [7]. This provides access control to different mass memory partitions held on Mass Memory modules, with the following partition types supported: File System, Packet Store, FIFO Buffer, Linked List and Raw Buffer. The Mass Memory modules are accessed using the SOIS services mapped onto the SpaceWire RMAP protocol [8].

- **fault-tolerance** – This implements an n+m redundancy approach to processing modules (n nominal processors and m redundant processors) rather than the traditional 2n approach. It relies upon and takes advantage of the other features of GenFAS – distributed software and universal access to all features using the SpaceWire network. OBSW Applications are linked together into a validated software build to be deployed together on a processing module. Each software build is stored within a build repository in the System Context area of Mass Memory. Upon start-up of the system, each software build is allocated by the GenFAS Configuration Manager to an available processing module of the appropriate type (it is assumed that there may be a small number of different types of processing modules, e.g. Leon2 and PowerPC), downloaded from Mass Memory and executed. Those processing modules not required to run the builds are held as a pool of spared. Should the processing module fail, the FDIR Manager will detect the failure and instruct the Configuration Manager to re-deploy the software build on one of the spare processing modules, i.e. a spare processing module is assigned to load the software build from Mass Memory and execute it.

4 FDIR ANALYSIS AND PERFORMANCE

The FDIR activity requires that the hardware provides facilities for the detection of faulty modules, containment of faults to a module and control over nominal and redundant functions. The details of the required level of fault monitoring are one of the main tasks in the initial phase of the project. The hardware will provide voltage level monitoring, watchdog timers and control to disable functions. The balance of software and autonomous hardware control of the system is an aspect that needs to be addressed within the project.

It is anticipated that the majority of the FDIR functionality will be implemented in software, where an adaptable decision making process can be supported. There should also be some hardware only FDIR mechanisms as it is anticipated that there will be credible failure modes that may stop processors and hence software from running. It is anticipated that there will be circumstances where it is desirable to stop all software from running and to diagnose and mitigate failures via TM/TC actions.

Two approaches will be considered for the FDIR architecture. One is a centralised architecture where the key FDIR mechanism, in particular these implemented in hardware shall be centralised and a classical prime and redundant hardware FDIR handling system is implemented with a prime and a redundant node. The other FDIR approach is a distributed one, where the FDIR handling mechanism is relocated on each node of every type. The distributed FDIR then relies on some arbitration or majority voting mechanism to reach consensus on the FDIR actions. It is anticipated that the software based FDIR will be distributed in either case.

The FDIR element of this project will be in two parts: the first is to develop an FDIR strategy which takes into account the modular architecture of the proposed system and the reliability characteristics built into the SpaceWire network standards. This work will culminate in an FDIR analysis tool which is intended to be implemented using the UML modelling language. The tool will include algorithms which can take as its input a definition of the system in terms of the type and number of nodes as well as how they are interconnected together via the SpaceWire network. The input will also specify the data paths and rates as well as constraints on commanding latency and priority. The tool will analyse the system and produce a table of FDIR actions which is used to define the conditions and paths of transitions for a state machine which shall be at the heart of the FDIR handling algorithm.

The FDIR handling algorithm which shall be implemented as part of the GenFAS software shall also be modelled in UML. The combined FDIR table generation, FDIR algorithm handling model and system definition shall be used to create a sophisticated FDIR model of the system which will enable the user to evaluate and optimise the system's architecture and performance when exposed to various scenarios of failure events.

As mentioned in the introduction, a major part of the analysis, and testing shall be focused on the reliability of the SpaceWire network when used within the MARC framework to handle the distribution of critical commands. An assessment shall be made to enable system architects to make a decision on whether such a system can rely on the SpaceWire network alone for distributing commands and their responses within a time critical embedded system.

5 REFERENCES

1. W.Gasti, “Advanced Robust Processing Architecture “ARPA” for Modular Architecture for Robust Computing “MARC””, TEC-ED/WG/2005-12, July 2006.

2. W.Gasti, “Generic Fault-tolerant Software using SOIS “GenFAS” for Modular Architecture for Robust Computing “MARC””, TEC-ED/WG/2005.14, July 2006.

3. “Software – Part 1: Principles and Requirements”, ECSS-E-40-1B, November 2003.

4. “Telemetry and Telecommand Packet Utilization”, ECSS-E-70-41A, January 2003.

5. “Spacecraft Onboard Interface Services – Informational Report”, CCSDS 850.0-G-1, Green Book, Issue 1.0, Washington, D.C, June 2007.

6. “SpaceWire – Links, Nodes Routers and Networks”, ECSS-E-50-12A, January 2003.

7. J.Stevens, D.Durrant, S.Fowell, “Generic Architectures for Future Mass Memories”, DASIA 2005 – Data Systems in Aerospace, Edinburgh, UK, May 2005.

8. “Remote Memory Access Protocol”, ECSS-E-50-11, Draft E, December 2005.