System-Level Mitigation of SEFIs in Data Handling Architectures, A Solution for Small Satellites

Mrs. Shazia Maqbool
Supervised By Dr. Craig I Underwood
Surrey Space Centre, University of Surrey, Guildford, Surrey, GU2 7XH, UK
s.maqbool@eim.surrey.ac.uk

Abstract – **We describe a fault-tolerant architecture designed to enhance commercial-off-the-shelf (COTS) device-based space-system reliability, and to provide automated system recovery, in the presence of radiation-induced functional errors. The architecture is primarily aimed at cost-effective "small satellite" systems, where very limited mass, volume and power resources preclude the use of multiple-redundant system-based architectures.**

Our architecture is based on the concept of a fast data network interlinking all units of the data handling subsystem to an intelligent supervisor node. The supervisor monitors status messages from the units and intervenes when the state of a unit does not match expectations or messages stop arriving. In such an event, the supervisor attempts to identify the nature of the fault and to recover the unit accordingly. Thus, this approach is flexible enough to support the fault-tolerant strategy deemed most suitable for the devices under consideration, given their failure modes and operating environments.

I. INTRODUCTION

One of the primary design considerations for a space mission is the survivability of its electronic systems in an ionizing radiation environment.

In the past, "rad-hard" technology derived from military programmes dominated the space industry. However, the decline in the availability of such technology and the trend towards ever more complex space missions has led to an increasing consideration of the use of state-of-the-art commercial-off-the-shelf (COTS) microelectronics for the construction of spacecraft systems. This is particularly the case at Surrey, where the design construction and operation of cost-effective "small satellite" missions typically requires the extensive use of current COTS technology to enable the spacecraft to achieve significant functionality in very severe power, mass, volume and cost constraints.

The use of modern COTS technology brings new issues with regard to system reliability in the space environment. By its very nature, COTS technology is constantly changing. One of the most significant trends is the move towards smaller dimensions – i.e. "scaling". Scaling has some benefits for radiation effects resilience: As gate oxide thicknesses decrease, hole trapping and interface trap build-up are becoming less significant, leading to a trend toward improved total-ionising dose (TID) performance [1,2]. Similarly, the increasing use of buried epitaxial substrates over the last 20 years [3,4] has helped decrease single-event latch-up (SEL) sensitivity – both by limiting the charge-collection volume, and by decreasing the substrate series resistance [5,6,7]. In addition, the concomitant trend towards reduced supply voltage levels suggests that device threshold voltages should soon fall below that required to sustain latch-up. Indeed, if continued scaling forces the industry to adopt silicon-on-insulator (SOI) technology, latch-up should soon be eliminated [1,4].

However, whilst some radiation effects are becoming less of an issue, new threats have emerged:

Another significant trend in COTS technology is move towards the use of "smart" logic, in device architectures, e.g. advanced memories, complex micro-controllers and field-programmable gate arrays (FPGAs), etc. Here the performance of the logic device requires the inclusion of complex control circuitry internal to the die. Whilst transparent to the user, such circuitry may offer a significant target to ionising particles and thus be prone to single-event effects (SEEs), such as a single-event upset (SEU) or single-event transient (SET). In such circuitry, these can manifest themselves as single event functional interrupts (SEFIs), whereby the device exhibits an unexpected change in its observable output state [8].

SEU in external memory is relatively easily mitigated by use of appropriate error-detection and correction (EDAC) coding strategies [9]. SET in external logic can similarly be dealt with by careful attention to system design and clocking circuits [10]. However such events internal to complex device structures remain a problem.

One approach is to build-in mitigation at device level (e.g. through the use of triple-modular redundancy within the device [11]) – however, this approach is rarely available in true "COTS" components. Instead, we must adopt a mitigation strategy that accepts that SEFIs can (and probably will) occur at device level, and design our systems accordingly.

The use of "n"-level redundancy, and "lock-step" processing remain powerful tools to mitigate such effects at system level [12], but the additional system overheads these methods entail (in terms of volume, mass and power) are always a problem in spaceflight. Indeed, they are particularly so in the context of the "micro/nano"-space systems designed at Surrey, where the entire spacecraft are typically only a few 10's of kg in mass. Also, it is worth mentioning that for longer spacecraft mission time frames, lockstep conditions for commercial devices must be well thought out. In particular, the TID degradation of the commercial devices must be examined for clock skew with increasing dosage. This may potentially cause "false" triggers if any device responds to dosage even slightly differently.

Instead, we propose a system architecture solution, where a single intelligent supervisor is added into the data-handling network. This monitors all the spacecraft systems and checks for unexpected changes or loss in functionality. This supervisor holds set-up and configuration data for the systems under its control, and it is continuously made aware of their current state by means of status messages passed to it by the systems.

Should an anomaly occur, the supervisor attempts to determine the nature of the fault and apply the appropriate recovery procedure. In this way, the supervisor acts as a kind of intelligent "operator in the sky", enabling autonomous recovery without the need for immediate ground intervention. Because of the question of "who guards the guards" the supervisor itself must be radiation-hardened – but this is not onerous given that it is a single unit which does not have to perform other processor intensive tasks. All the other sub-systems on the OBDH network can be advanced COTS based.

To offer an "intelligent" mitigation strategy, we must be aware of the nature of SEFIs in the device types under consideration and their signatures:

II. Functional Interrupts in Typical COTS Data Handling Devices

A. Memories

With regard to SEFIs, floating-gate memories and DRAMs are of particular concern.

With floating gate memory technology, every bit is represented by a metal oxide semiconductor (MOS) transistor with two gates – the normal MOS gate, and the "floating" gate, which is surrounded by high-impedance insulating material. Charge stored on this floating gate represents the bit state, and this remains even if the device is powered-down. The technology may be used to construct electrically erasable programmable read-only memory (EEPROM) or so-called "flash" memory.

In EEPROM devices, electrical erasure of individual stored bits is made possible by applying a voltage of the opposite polarity to the charging voltage to the non-floating gate. Large EEPROMs allow erasing only in fixed-size blocks, typically 128K-bits – 512K-bits at a time, hence making the erasure process faster. These are called "flash" memories.

In contrast to the read operation, writing to floating gate memories is a slow process. Therefore, in order to provide improved performance, flash devices are organized in blocks and pages. An embedded state machine is used to control the flow of data to-and-from the device and this uses page buffers to hold data temporarily during any read/write operation. It also issues internal commands and controls sequencing to perform read, write or erase operations. Some enhanced devices also include the ability to queue a sequence of commands and provide automatic power saving features – all of which requires internal control logic.

Flash devices can be operated in different modes and the visibility of internal changes caused by radiation effects depends on the way that the part is used [13,14].

Radiation testing of these devices has revealed their high susceptibility to SEFIs. Table I summarizes SEFI signatures that have been observed in flash memories along with proposed recovery procedures [13,15,16].

A dynamic random access memory (DRAM) cell stores information on a tiny capacitor accessed through a MOS transistor. Refresh cycles are used to update every memory cell periodically. In order to make the refreshing task simple and manageable, DRAMs are organized in two-dimensional arrays (i.e. cells are organised in rows and columns). Larger DRAMs often have multiple arrays and this eases the electrical and physical design problems that would otherwise occur with an extremely large array. Another advantage of multiple arrays is the parallelism that can be achieved, enabling a modern DRAM controller to perform several operations at once. For example, it can complete a write operation in one array whilst initiating a read operation in another, thus increasing the effective throughput of the memory [17]. Again this requires internal logic.

The existence of SEFI in DRAMs was first reported in 1996, where during irradiation, a DRAM device was observed to enter its test/standby mode [18]. Reading the device during this mode results in an unexpectedly high "upset" rate. The device tested in this example was a Samsung 16Mbit DRAM. Similar signatures were observed during irradiation of an Oki Semi 4Mbit DRAM [8].

TABLE I
SEFIs IN FLASH MEMORIES

SEFI Type	Manufacturer	Recovery Method
Read operation locked into an endless loop, with an increase in supply current	AeroFlex	Power cycling
Read operation locked	AeroFlex, Toshiba, SanDisk, Intel, Samsung	Repeat the read process or cycle power
Write operation locked	Intel, Samsung	Power cycling
Row/column changes: Large portions of the memory array change state	Intel	Power cycling
Block-erase: the device is stuck in a "busy" state	AeroFlex, Toshiba, SanDisk, Intel, Samsung	Power cycling
Partial-erase: the device requires repeated erase commands for one or more blocks	AeroFlex, Toshiba, SanDisk, Intel	Repeat the erase operation

DRAM SEFIs were first observed in orbit in a 12 Gbit solid-state data recorder (SSDR) installed on the Hubble Space Telescope (HST) [19]. As with many DRAM devices, the HST DRAMs include redundant memory rows and columns. When the device is powered up, weak rows/columns (i.e. those where the data retention of the cells is suspect) or bad rows/columns (i.e. those where bits are always incorrect) are replaced with redundant ones, as appropriate. An internal redundancy latch holds the device configuration. It is believed that upset in this logic element led to the observed "block" SEFIs, caused by correct columns/rows being replaced by bad or week ones.

Makihara also reports block SEFIs in NEC 16 Mbit DRAMs [20]. The author suggests that these errors might be due to upsets in the control circuitry turning on the gates of access transistors before the bit lines are pre-charged, resulting in unknown data being stored into the cells.

Synchronous DRAM (SDRAM) technology represents the state of the art in high density, volatile memory. These devices have internal state machines to provide pipelines, programmable refresh modes and power control states, etc. A mode register is included on chip, which is used to define the device operation.

Usually the mode register is configured once, when the device is powered-up. This register has been found to be susceptible to SEE in some devices [21,22].

Particular bit combinations in the mode register are not defined or are reserved for future use. Thus, by producing one of these patterns, a single particle-strike can result in complete loss of functionality of the device until the device is reset or, in some cases, until power is cycled.

In a SDRAM, the memory array is divided into two or more banks. This allows one bank to be pre-charged whilst the other is being accessed. The resulting parallelism in operations can provide improved throughput. However, a SEU in the control circuit can cause block errors when the active bank information is lost [21].

Row-based errors have also been observed [22]. Typically, a large number of errors were seen in sequential rows, i.e. several rows that were next to each other are seen to have a similar number of errors.

Several types of the "logic" SEFIs are also observed, sometimes accompanied by an increase in standby current of approximately 0.5mA to 2mA.

In most of cases, a device can be recovered from a SEFI by reinitializing affected device and rewriting its mode register. Probability of SEFIs occurrence can also be reduced by periodically rewriting its mode register[23,24].

B. *Field Programmable Gate Arrays (FPGAs)*

There is growing interest in the use of re-programmable FPGA technology for space missions.

SEUs in these devices can be grouped into three categories [25]:

Configuration upsets are defined as the upsets in the configuration memory of the device and can be detected by read-back. The likelihood of failure will depend upon the upset location and the specific design utilisation of the device resources. It is reported that on average 6.5 configuration upsets are required to produce a functional failure in a device with no mitigation at all [26]. A configuration upset can sometimes lead to high current states. For example, a SEU may cause two output drivers to be connected together [27].

User logic upsets represent upsets in the circuit elements that are not directly accessible by read-back. The effect of these is dependent upon the particular logical design implemented by the user.

Architectural upsets are those upsets that occur in the control circuitry of the FPGA. As an example, it is possible for a single upset in the configuration control circuitry to change many configuration bits simultaneously [26]. This kind of anomaly can only be detected by noting the malfunction of the device. Mattsson reports such an event, whereby all shift registers are disabled at the same time. The device requires a reset and complete reconfiguration for recovery [28].

C. Microprocessors

Microprocessors are often sensitive to SEUs, which can occur in any basic element of the processor including the program counter, the sequence controller, the registers, and the arithmetic logic unit (ALU), etc.

Depending on time and location of the upset, it may manifest itself as a calculation error or a SEFI caused by alteration of control or address bits. In many cases, the error occurs in unused elements and thus has no observable consequences.

Errors have been observed where the processor simply resets itself or stops functioning altogether. The latter is called "lock-up" SEFI (or simply lock-up) [8]. Hence, a microprocessor SEFI may cause the processor to lockup, reset, have continuous exceptions, execute its standby mode, or go into some unknown state.

D. Data Handling Networks

Commercial data handling networks are becoming of increasing interest for space missions. For instance, European space agency (ESA)/European space research and technology centre (ESTEC) program European cooperation on space standards (ECSS) has recommended IEEE 1355 (SpaceWire), whilst national aeronautics and space administration (NASA)/jet propulsion laboratory (JPL) X2000 program has adopted the IEEE 1394 (FireWire) network standard.

Results on radiation testing of both spacewire and firewire have been published in Ref[29, 30, 31]. Broadly, errors can be divided into two categories: soft errors, which do not disrupt data transmission across the network and hard errors or SEFIs, which stop communication and are most likely to be caused by an upset in protocol registers or control logic. Chau et al. describes the most common or critical failure modes for spacecraft data handling buses, which are being identified by JPL/NASA [32] and are summarized below

- Invalid Packets: These are the packets, which have invalid data.
- Non-Responsive packets: An anticipated response to a message does not occur before it times-out.
- Babbling: Communication among nodes is blocked or interrupted by uncontrolled data stream.
- Conflict of Node Addresses: More than one node has the same identification.

Buchner and Rodriguez presents their results of pulsed laser testing of an Atmel ASIC chip, which was used to generate the spacewire interface. Three types of errors were observed:

- data errors alone
- loss of link alone
- data errors together with loss of link

Recovery involved restarting of the protocol software. One SEFI was observed, which required power reboot of the network [31].

III. System Level mitigation of functional interrupts

SEFI's, then, are an inherent feature of the types of COTS devices we would wish to use to construct cost-effective and capable space systems. Whilst device cross-sections are usually relatively low, and thus such errors are not expected to occur all that frequently in the space environment, none-the-less the consequences of SEFIs are important for system reliability.

In contrast to traditional single-event upsets (SEUs), where faulty bit can be isolated and corrected, SEFIs occur in the sensitive cross-section of the device to which the user has no direct access or has minimal information about device architecture. Therefore, the exact location of the fault in the device is unknown and it can only be detected by the noted malfunction of the affected device. This implies that each module in the data handling architecture should be provided with some kind of intelligent monitor to detect SEFI like signatures as quickly as possible. However, providing this at module level is undesirable as:

- it will require at least one radiation hardened core within each module;
- potentially, a major design modification would be required on each module.

We therefore propose an architecture whereby all systems are linked to a fast data bus, which is itself linked to an intelligent "supervisor" module. The supervisor requires fast communication to and from the underlying devices to track their activities. We exploit the fact that there is already a move towards adaptation of fast data handling networks for space missions. A multi-level fault protection technique has already been demonstrated by Chau et. al [32] to provide high speed data handling for highly reliable space systems. This brings the opportunity to centralize the desired intelligence and autonomy into one rad-hard unit to supervise the on-board data handling architecture. This technique is a generic solution for providing fault tolerance in a data handling architecture, which is heavily based on commercial components and standards. However, it will be particularly useful for small satellites because of low overheads associated with this approach. Thus, the purpose of this work is to investigate feasibility of an on-board intelligent supervisory system to detect and recover from gross failures, such as SEFIs in an otherwise COTS-module-based spacecraft data handling architecture.

System availability requirements address extreme events leading to possible loss of mission as well as less severe events, which might require ground station. Usually small satellites require ground-intervention to recover from events like SEFIs, resulting in long downtimes. An intelligent on-board operator is desirable for future missions in order to meet increasing demand of maximum system availability.

Also, this approach promises high adaptability and reusability. The target system will be able to provide a nominal operation without the supervisor node, if the mission does not require achieving a certain level of reliability and availability. On the other hand, inclusion of the supervisor node will lead to increased reliability and autonomy with minimum ground intervention. The overheads associated with this approach come in the form of the level of complexity of the software running on the supervisor and network traffic, which can always be traded according to the mission requirements, even without making any change to the system hardware. Therefore, it promises a measurable increase in small satellite utility across range of mission performance requirements.

IV. Example Data Handling Architecture

An example architecture based on the proposed principle is shown in Figure 1. Each system links to the network via an interface gate-array, which also provides for local monitoring and mitigation of radiation effects. In principle, the data network could be any reasonably fast bus, however, we favour the use of the newly-emerging SpaceWire (modified IEEE 1355) network technology [33].

A. The SpaceWire Network

SpaceWire provides for very fast data transfer (a minimum of 2 Mbps, with a capability up to 155 Mbps using low voltage differential signal (LVDS) drivers) at low power (~5mW/Mbps @ 100Mbps), scalable, i.e. network bandwidth can be increased by adding nodes, and provides a high degree of isolation between systems – avoiding problems such as powering up from interconnections between systems etc.

JPL have similarly identified SpaceWire, along with the IEEE 1394 FireWire to be of interest for their X2000 COTS-technology based data handling architecture [34].

SpaceWire is a full-duplex serial point-to-point network, in which nodes are interconnected by routing-switches. The SpaceWire standard defines six layers of protocol: *physical, signal, character, exchange, packet* and *network*. It provides a coherent interface to processors, mass memory units and sensors etc. Whilst there are similarities to the IEEE 1394 bus standard at the physical layer, the higher layers of the SpaceWire protocol are much simplified in comparison. This makes implementation easier (a basic SpaceWire link can be implemented in 5000 gates), acceptable overheads on even smaller packets (<35%), but it does preclude some useful functions – such as the ability to support broadcast or multicast.

The spacewire standard describes the hardware and software necessary for implementation of the protocol. The standard includes useful fault tolerance features such as parity checks, link disconnect error detection and recovery, escape sequence error, credit error, empty packet error, exception end of packet received and destination address error. It also specifies a very small underlying bit error rate, being less than 10^{-14}, which means one error every 11.5 days at 100Mbps. However, radiation response including SEUs and total dose effects would depend on the protocol implementation chips. For instance, an Atmel ASIC-based spacewire router is reported to have a total dose performance of 300krad, and SEU and latchup immunity up to 100MeV[35].

In our proposed implementation, the network is used for sending status messages, diagnostic commands and blocks of data (e.g. payload data) between systems. A separate Controller Area Network (CAN) bus is used to provide a redundant command and configuration control function. The system topology is shown in Figure 1.

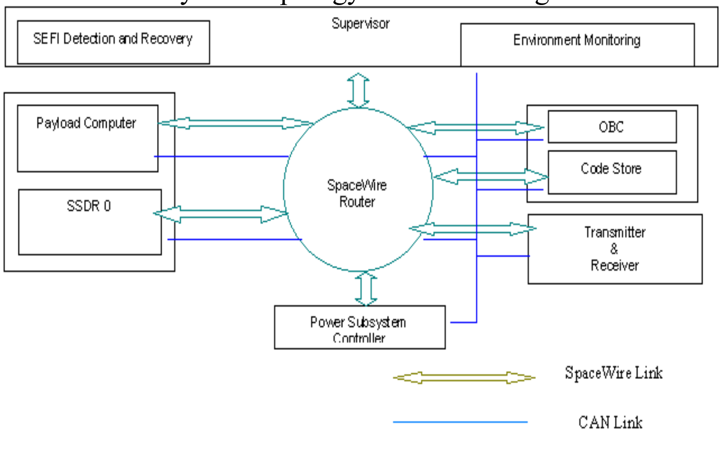

Fig. 1. Example OBDH Architecture

The data handling network is vital to success of a space mission. Therefore it should be sufficiently radiation tolerant. Actel Corporation offers two classes of rad-hard FPGAs, RH1020 and RH1280 with equivalent gate densities of 2,000 and 8,000 respectively. These products are latchup immune and can withstand total dose in excess of 300Krads (Si). Its SEU performance is predicted to be 10^{-6} upsets per bit-day in a 90% worst-case geosynchronous earth orbit. Actel proposes mitigation techniques to improve SEU performance of these devices by incorporating triple-modular redundancy (TMR) or by avoiding flip-flops in the sequential modules [36]. Adaptation of one of these techniques is recommended for critical design sections.

Another rad-hard FPGA RHAX250-S is to be launched in a couple of years [37]. This device technology offers a gate count of 250, 000 with even better radiation performance.

In addition to these rad-hard products, Actel offers radiation tolerant devices such as RTAX_S. This device technology has TID response comparable to RH1020 and RH1280 devices, and latchup and SEU immunity to 104 and 60MeV-cm^2/mg respectively [38].

Depending on mission requirements, a rad-hard or rad-tolerant FPGA could be adopted to act as the interface FPGA. Implementation of the supervisor will be discussed in proceeding section.

B. The System Nodes

The systems shown provide the core OBDH functions of the spacecraft, and comprise different device technologies. For example, the on-board computer (OBC) (Figure 2) is based around a commercial micro-controller, and has built-in program storage memory protected by a hardware-EDAC coding scheme such as the (16,8) double-bit error correcting code used on previous Surrey spacecraft.

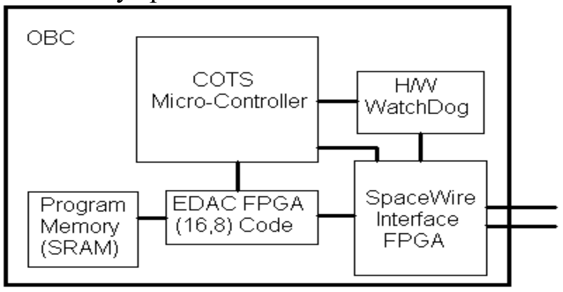

Fig. 2. The On-Board Computer (OBC) System Node

The interface FPGA will "wash" the memory to prevent the accumulation of bit errors, and error statistics will be gathered for passing on to the external supervisor module. The supervisor can monitor and recover the OBC via the interface FPGA, which has access to the memory and processor. If necessary, it can reset the processor and ultimately, it can power-cycle the entire OBC via its link to the power system controller (Figure 3.)

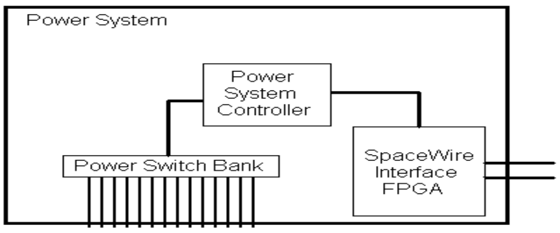

Fig. 3. The Power System Node

The code store (Figure 4) will be based on floating gate memory and will hold the back-up code, and the operating system for the OBC, as well as other critical data such as the current state of the spacecraft's systems, and the configurations for any reconfigurable FPGAs.

The supervisor will use this information to recover the spacecraft from system malfunction, e.g. by re-booting the OBC or payload computer, resetting device configurations, and setting the state of control variables based on their last known good state.

Because of the critical role that the code store will play in any system recovery, TMR is used within this system to guard against SEFIs in the floating gate memories. Given the small physical size and low power consumption of the Flash devices, this is not onerous in terms of system resources. Indeed, the TMR voting logic may well be implemented directly in the interface FPGA.

The flash memory cells can suffer from loss of charge when subjected to heavy ion irradiations [39]. The memories will be periodically checked for errors. If the upset rate is higher than a threshold value, it will be treated as a SEFI event, the supervisor can cycle the power to the affected memory and restore its contents. For the upsets lower than a threshold, correct data will be written to the corrupted cells. Keeping record of the erroneous locations can further increase detection performance of the system. This is because of the reason that an affected flash memory cell can again discharge itself because of the radiation-induced traps in floating gate oxide [30]. In this case, a permanently damaged cell would require to be removed from logical memory map.

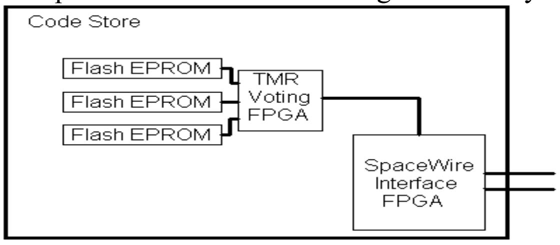

Fig. 4. The Code Store Node

The payload computer uses a reconfigurable static-RAM (SRAM)-based FPGA for its processing element

(Figure 5). This allows the payload computer's hardware to be optimised to match the particular requirements of the payload at any given time. The configuration data for the FPGA will be stored locally in non-volatile ferro-electric RAM (FRAM), with a copy held in the code store. The FPGA processor will have its own in-built EDAC circuit for protecting the program memory, and again error-rate statistics will be made available to the external supervisor via the interface FPGA. The interface FPGA will be responsible for read-back to check for configuration errors under direction from the supervisor.

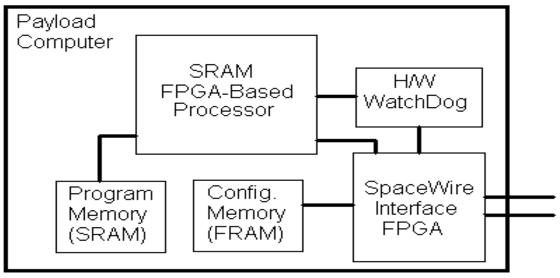

Fig. 5. The Payload Computer Node

SSDR0 and SSDR1 represent a pair of solid-state data recorders, comprising DRAM or SDRAM devices. The memories will be organised in banks with separate power switches, so that in the event that power has to be cycled through a device, only a proportion of data will be lost.

As with previous Surrey satellites, these bulk memories will be protected by Reed-Solomon EDAC codes (Figure 6). Again error rate statistics will be gathered by the interface FPGA.

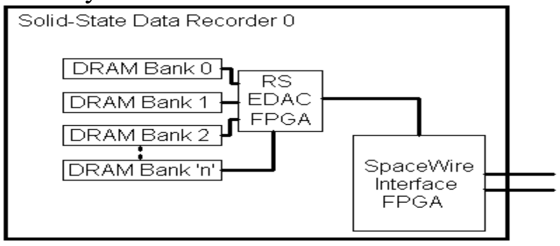

Fig. 6. A Solid-State Data Recorder Node

The Supervisor

The supervisor is the key to this architecture. It provides the functions of system monitor/fault detection, fault diagnosis, and system recovery.

It is important then that the supervisor is robust against radiation effects (Figure 7). A rad-hard microprocessor will be adopted to act as an intelligent supervisor. Probability of getting a functional interrupt in the supervisor node will therefore be quite low. Inclusion of watchdog timers will ensure recovery of the supervisor from functional interrupts, if any occurs. Nevertheless, the supervisor's role is merely a monitoring entity. The data handling architecture will continue to operate with a reasonable reliability because of the mitigations adopted within each module for down time of the supervisor. In a worst case scenario, an exception in the supervisor can lead to false triggering of the fault conditions on one or more modules. In order to cope with this possible failure mode, the supervisor will be provided with a threshold of recovery attempts to each underlying module, once it reaches that threshold the corresponding program will go into sleep state until ground intervention restarts it. However, this situation is not likely to occur very often.

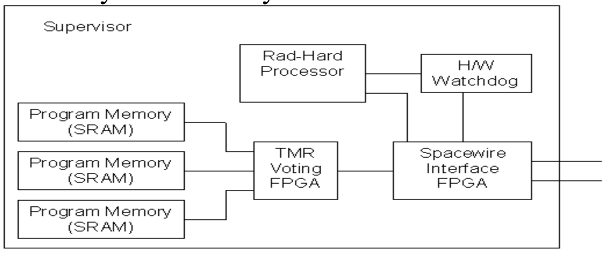

Fig 7 Supervisor Node

Current Protecting Power Switch (CPPS)

As devices are likely to experience changes in their current consumption when affected by a SEFI, it is desirable to provide current monitoring and protection. Next to spacewire interface FPGA, CPPS forms part of the interface node on each module.

V. SUPERVSIORY FUNCTIONS

It is important to note that the supervisor will be monitoring sub-systems with different classes of device – e.g. microprocessors, DRAMs, floating gate memories, FPGAs etc. - which may all exhibit several different signatures as the result of a SEFI. For example, a SEFI in a microprocessor may cause it to "hang", reset, have continuous exceptions, execute its standby mode, or go into some unknown, unrecognizable state. DRAMs, may show block errors or the execution of undefined states. Floating gate memories can have a wide range of signatures including repeated errors in the same word, all address locations in error, read or write operations entering an endless loop, large arrays of memory changing state, etc. and FPGAs have a similar complex set of signatures. In addition, supply-current variations may be seen in all these devices.

Recovery strategies will also be different in each case, and the priority accorded to different signatures might also differ. For example, if a device halts with a sharp increase in current, this might indicate that device has entered a potentially damaging micro-latch state and in this case immediate power cycling through the affected device may be required.

This all implies that the supervisor should have sufficient intelligence and adaptability to change the fault tolerance strategy dynamically based on the device type, failure mode, and the operating environment (figure 8).

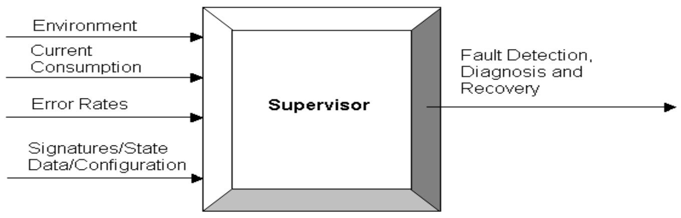
Environment
Current Consumption
Error Rates
Signatures/State Data/Configuration
Supervisor
Fault Detection, Diagnosis and Recovery

Fig. 8. Supervisor Inputs and Output

VI. Monitoring/ Fault Detection

The system nodes are either intelligent (containing a microprocessor within the module) or non intelligent (memory units). The supervisor consists of a set of programs, one corresponding to each subsystem. We have adopted user datagram protocol (UDP) to access a particular program. We favour use of the open standard rather than custom implementation as it offers seamless interconnectivity between spacecraft and ground, ease of implementation, increased error protection and promotes compatibility of interfaces.

Intelligent Nodes

The supervisor must offer intelligent monitoring of the systems on the network. It does this by keeping track of the tasks running in any of the systems attached to the network. As soon as a system begins a task, it will send a "start-of-task" packet to the supervisor.

Idea of this packet is that it will contain information about the specific task running on the machine, any diagnostic data, which can indicate health of the machine, any state data, which can be useful for recovering state of the machine as close as possible to the one prior to fault event. This information will depend on the machine under consideration and tasks performed by it. This proposal is based on the fact that SEFIs detection requires careful monitoring of the device behaviour and its recovery usually involves resetting and power cycling of the affected device and therefore it is important to have any state data. It emphasizes the fact that the designer should carefully choose parameters to reflect the true behaviour of the device. This paper presents example parameters for an OBC unit.

Considering a typical multiprogramming environment, where all the software including the operating system is organized into a number of processes that can execute in parallel. Each process is loaded into EDAC protected memory as a separate executable file. One of the processes will be a special process. This process will copy contents of the program memory into code store at a location, known to the supervisor. Typically, an operating system allows each process to run for (10-100) ms before switching to the next process. Considering a total of 20 processes, and a round robin scheduling scheme, this process will update state of the OBC every (0.2-2)s Even with a priority-based scheduling scheme, this statistics will remain more or less the same. As most of the processes are run with the same priority to avoid situations where some processes will never be executed.

The interface FPGA receives a start of task marker, every time a process is switched from the ready to the running state. The interface FPGA packetises it and sends it to the supervisor. The packet format is shown in the Fig 9.

System ID	Packet Length	Task ID	Flags	Diagnostic Health Data

Fig 9 The Packet Format

System ID informs the supervisor that is it a packet from the OBC processor or from the OBC interface FPGA. Task ID is ID of the task currently running on the OBC. Diagnostic health data (DHD) contains the EDAC error count. If it was a test task, it will hold results from that. The packet also includes a screech flag. A screech is a report of an unusual error or fatal program trap from the operating system running on the OBC machine**.** Summary of the flags field is presented in the table II.

Table II

Description of the Flags field

Flags	Description
Bit 0	I am okay packet
Bit 1	Screech packet
Bit 2	SEU count to be found in DHD field
Bit 3	Test task results are to be found in DHD filed

The current protection circuitry is set to a default threshold current value, when it receives the start of task marker from the OBC.

The supervisor software consists of a detection/diagnostic and recovery program for each underlying module. Each program is further divided into two parts corresponding to module itself and its interface FPGA. The detection/diagnostic table entries for the OBC module are shown in the Fig 10.

Task ID	Current Consumption	Diagnostic Health Data	Storage for Screech Packets

Figure 10 Parameters Corresponding to the OBC Module

The designer is required to provide a maximum possible current consumption corresponding to each process. Once, the supervisor receives a packet it sends back a packet to the interface FPGA with a threshold current value. After receiving this packet, the interface FPGA changes default current threshold in current protection circuitry to this new value.

The supervisor will expect to receive next start of task packet before it times-out (100ms + an additive factor corresponding to network delays). If the message is not received and the timer times-out then the supervisor will begin its error-recovery procedure for the OBC.

A similar approach can be adopted for payload computer. A packet time-out/non responsive FPGA will be indicative of an architectural upset or configuration upsets. Mismatch of test program results can be caused by configuration or user logic upsets. Critical section of user logic can be guarded by periodically reading back for comparison with stored values or by periodically rewriting these values.

Non-Intelligent Nodes

For systems that lack a processor (and therefore are not naturally executing tasks), or for idle systems, the interfaces themselves will run a system-monitoring task, involving, for example, the washing of memories to establish SEU-rates, etc. The supervisor will periodically poll the SSDRs to collect these error rates and will use these to detect a SEFI event.

For instance, Guertin *et al* presents dynamic SEFI detection and recovery for ground testing of Hynix/Hyundai SDRAMs devices. They define a SEFI as an event when n out of N bits in a memory region are in error. They have calculated a n:N ratio of 0.375, i.e a SEFI is declared when 96 out of 256 bits are in error or 348 in 1024 and so on [24].

While calculating a threshold for a space mission, one needs to take into account orbit and environment, which is to be encountered by the spacecraft.

Network Considerations

Fault tolerance in data handling network forms an integral requirement for success of this approach. And therefore need to be taken into account. Section II D describes most common failure modes of data handling networks. This paper suggests possible remedies in context of this example architecture.

The supervisory approach expects all units/modules on the bus to periodically communicate to the supervisor. The supervisor waits for packets from intelligent nodes and it polls non intelligent nodes. Any communication on the SpaceWire requires both source and destination to perform hand shaking before sending any data. In addition, both ends are required to send nulls to keep the link alive during a communication session (when no useful data is being transferred). Therefore, SpaceWire is capable of detecting if any of the source or destination got disconnected for any reason. Active end tries to re-establish the link. If it cannot, it disconnects itself from that link. It can be programmed to report application layer of any problem encountered. In this way, the supervisor will know, if any of the nodes is not responding. If a node other than the supervisor detects a non-responding node (including non-responding supervisor node), it will send node ID to the supervisor on CAN bus. So that the supervisor can initiate a recovery procedure on the non-responding node.

SpaceWire can detect parity error, escape sequence error, credit error, empty packet error, exception end of packet received and destination address error. Adaptation of IP/UDP standard further increases error check by inclusion of checksums. Therefore, any invalid packet can be discarded.

A babbling node is a situation where a node sends out small packets in an infinite loop in order to check for some information such as status reports. Babbling can lead to uncontrollable data stream in a bus network and can cause congestion at router in a router-based architecture. The first line of defense against anomalies like babbling and two nodes appearing with same address will be the inherent behaviour of the supervisory approach. Both of these anomalies are like to affect packet traffic to/from the supervisor.

For instance, if the supervisor does not receive an "I am okay" packet from the OBC and it times out, it will first poll the interface FPGA. If it cannot get connected to the interface FPGA, it will be considered as non-responding node, babbling or two nodes appearing with same address anomalies. And the supervisor will start its diagnostic and recovery on this node via CAN bus.

Also, packets more than expected from a node within a given time slot or packets with unexpected lengths can also be treated as fault indications.

Network Delays

The SpaceWire standard specifies a maximum delay of 5μs for 10 bits at a speed of 2Mbps [40], leading to a maximum delay of 5μs for 500bits, which is determined to be maximum possible length of a packet from the OBC to the supervisor, at a speed of 100Mbps. The SpaceWire standard does not specify any arbitration scheme for the router. It can be round robin, priority or weighted priority etc. For the system-level supervisory approach, a priority-based arbitration can assure that a timer in the supervisor will not time out because of routing delays.

Environment Monitoring

As well as monitoring the systems on the network, the supervisor will have access to a radiation environment monitor such as Surrey's CRE/CEDEX payload [41, 42]. These instruments are capable of producing counts of the incident particles on the spacecraft electronics. When the count rate exceeds some pre-set threshold, a flag is raised that informs the supervisor that the environment is hostile.

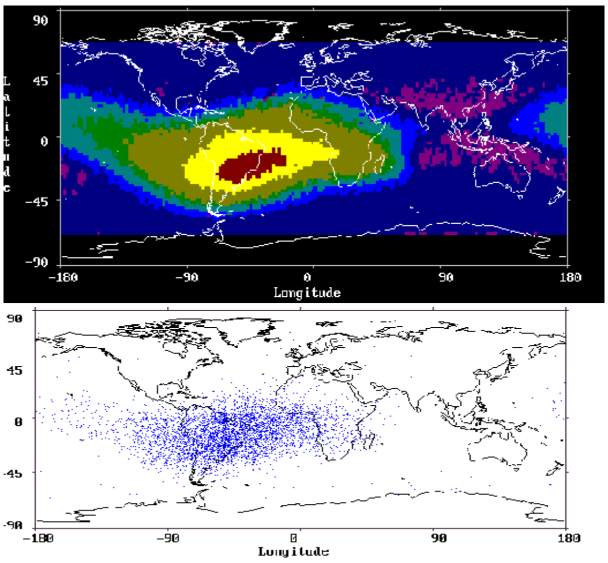

Fig. 11. (top) Proton Environment Measured by the *KITSAT-1* Cosmic-Ray Experiment (CRE) at 1330 km Altitude.

(bottom) *S80/T* Program Memory Upsets at 1330 km Altitude

For a satellite in low-Earth orbit (LEO), the probability of getting an SEE may be much higher when traversing the South-Atlantic Anomaly (SAA) (Figure 10), or when crossing the polar-regions during a solar particle event. Similarly, for a satellite in geostationary Earth-orbit (GEO), a normally benign SEE environment might suddenly become much more severe due to a solar particle event. Thus, real-time knowledge of the environment can enable the supervisor to take special, adaptive, measures in harsh radiation conditions. Device health checks can be made more often (e.g. increased washing of memories) and other precautionary measures can be taken (e.g. the avoidance of read/write operations on flash memories).

Fault Diagnosis

In addition to regular monitoring of system behaviour, the supervisor will periodically instigate further diagnostics by running tests. The type of test employed depends upon the devices used in the system. Diagnostic can also be enabled after detecting an erroneous behaviour.

For processors, a small test program can be run. The results of the test will be compared with the stored results and a recovery procedure will be invoked on mismatch of the results.

In addition to test programs, configuration of SRAM FPGA devices can be read back and checked against the configuration data stored in the code store.

In network interfaces erroneous behaviour is usually caused by an upset in protocol registers or state machines. For instance, Seidleck et al presents an example of a soft error in asynchronous request filter low (ARFL) register, which is a link layer register of IEEE 1394 (firwire) bus [30]. The function of this register is to enable reception of asynchronous packets. When an asynchronous request packet is received, the bit corresponding to the source ID is examined in ARFL register. If a SEU turns a bit from 1 to 0, packets with corresponding source ID will neither be acknowledged nor be queued. Similarly, the SpaceWire standard specifies a state machine to hold a sequence of steps for recovering from link disconnect errors. A SEU on one of the entries can cause erroneous behaviour and it will not be evident unless the node experience a link disconnects error.

As these elements are implemented as user logic and can be accessed by the programmer. Therefore it can be a useful measure to read these values periodically and check against values stored in TMR protected code store. In addition to packets described in preceding sections, each interface FPGA will send its critical parameters to the supervisor.

For memory devices, the output of the EDAC coding systems can be used to identify the address locations of errors and thus the faulty device or devices can be identified. If necessary, test bit patterns can be written and read back. If a faulty device exhibits permanent failure, it can be removed from logical memory map.

System Recovery

Once the fault has been identified, the appropriate recovery method can be invoked. After restoring basic functionality of the system, the supervisor can use data stored in the code store to attempt recovery of the state of the system to that prior to the fault occurring.

In the case of the OBC, for example, this might mean re-loading the flight software, and process context table, stack, queues etc., which were written to the code store in last known good state.

A similar approach would be taken for the payload computer, although here the internal configuration of the processor FPGA would also need restoring.

TABLE III

SUMMARY OF FAULT DETECTION, DIAGNOSTIC AND RECOVERY

Module	Detection	Diagnostic	Detection Capabilities	Recovery
OBC	1. Packet Timeout 2. SEU count of program memory exceeded the threshold 3. Screech Flag High	1. Test program results do not match expectation	1. Processor malfunctioning 2. Memory malfunctioning 3. Screech 4. Task Crash 5. Processor hang/lockup	1.Reset or power cycling with state recovery 2. Power cycling with state recovery 3. Reload task and report to ground 4. Processor reset 5. Power cycling and state recovery
Payload Computer	Same as the OBC	In addition to test programs, configuration read back	Same as the OBC + errors in configuration	Same as the OBC + configuring FPGA when required
SSDR	SEUs count higher than a threshold	Isolate affected memory bank	Upset rate indicating a SEFI	Copy contents of the affected memory device/bank to a redundant one and reinitializiation of the affected device or power cycling through the bank
Code Store	1. Upset rate higher than a threshold 2. Repeated errors	Isolate location of repeated errors	Upsets indicating a SEFI or a permanent error	Reset or power cycling. Report of permanent errors
All Interface FPGAs	1. Unable to connect 2. Link disconnect 3. Error reports received via CAN bus 4. More than expected packets received from a node 5. Packets with unexpected lengths	Diagnostic packets	1. Non responding node 2. Erroneous node 3. Babbling node 4. Node with incorrect source address	Reset command via CAN or reloading interface FPGA with protocol registers

Bulk memory systems such as the SSDRs would inevitably lose data, however, provided the individual banks of memory are not too large, the proportion of data lost can be kept low. If a SEFI event requires power cycling of a particular memory bank, then data from unaffected devices within the memory bank may be recovered and written to spare memory locations in other banks before the power is cycled.

With judicious choice of EDAC coding strategy, and by spreading logical blocks of data over several physical banks of memory, it may be possible to completely restore the original data

VII. CONCLUSION AND FUTURE WORK

Over the last 20 years, COTS technology has been flown in space systems with a reasonable expectation of success – as exemplified by Surrey's 20+ COTS-based space missions. However, the complexity of COTS device technology has been increasing and functional interrupts have become serious threats to reliable COTS based on-board computing. Detection of SEFIs requires intelligent monitoring of the device behaviour. Recovery procedures (there can be more than one) depend on the cause of the problem and its severity. The question arises how to recover the affected device quickly and as close as possible to its state prior to the fault event.

In past reports of SEFIs were confined to microprocessors only. However, this is no longer the case. SEFIs have been extended to memories as well as FPGAs. Power cycling requirements associated with most of the functional interrupts demands for an external entity to enforce state recovery. Therefore, it seems reasonable to apply fault detection and recovery at the system level.

In contrast to traditional mitigation strategies for SEFIs, which are heavily based on large amount of redundancy. It will be more efficient in terms of required resources to explore possible malfunctioning behavior of a target application and to design a fault tolerant system to cope with possible faults.

This paper has described a proposed autonomous on-board fault detection and recovery architecture designed to maximize system availability and information integrity in presence of functional interrupts.

Work is in progress to implement this architecture.

VIII. References

[1] A.H. Johnston, "Radiation effects in advanced microelectronis technologies," IEEE Trans. on Nucl. Sci., vol. 45, no. 3, pp. 1339-1354, June 1998

[2] Paul V. Dressendorfer "Basic mechanisms for the new millennium," IEEE NSREC Short Course, July 1998

[3] Johnston, A.H., "Scaling and technology issues for soft error rates," Paper presented at the 4th Annual Research Conference on Reliability, Stanford University, October 2000

[4] Oldham, T.R., "How device scaling affects single event effects sensitivity," IEEE NSREC Short Course, July 2003

[5] J.E. Schroeder, A. Ochoa, and P.V. Dressendorfer, "Latchup elimination in bulk CMOS LSI circuits," IEEE Trans. on Nucl. Sci., vol.27, pp. 1735–1738, December 1980

[6] A. Ochoa, Jr., and P.V. Dressendorfer, "A discussion of the role distributed effects in latchup," IEEE Trans. on Nucl. Sci., vol. 28, pp. 4292–4294, Dec. 1981

[7] P.E. Dodd, M.R Shaneyfelt, E. Fuller, J.C Pickel, F.W. Sextan and P.S. Winokur "Impact of substrate thickness on single-event effects in integrated circuits," IEEE Trans. on Nucl. Sci., vol. 48, no. 6, pp. 1865-1871, December 2001

[8] Koga, R. et al., "Single Event Functional Interrupt (SEFI) Sensitivity in Microcircuits," Proceedings of the 4th European Conference on Radiation and its Effects on Components and Systems (RADECS), pp. 311-318, 1997

[9] K. Label and M.M.Gates, "Single-event-effect mitigation from a system perspective," IEEE Trans. on Nucl. Sci., vol. 43, no. 2, pp. 654-660, April 1996

[10] David G. Mavis and Paul H. Eaton, "SEU and SET mitigation techniques for FPGA circuit and configuration bit storage design," Proceedings MAPLD Conference, 2000

[11] F. Faccio, "COTS for the LHC radiation environment: the rules of the game," Paper presented at the 6th Workshop on Electronics for LHC Experiments, Krakow, Poland, 2000

[12] P. Agrawal, "Fault tolerance in multiprocessor systems without dedicated redundancy," IEEE Trans. on Computer, vol. 37, no. 3, pp 358-362, March 1988

[13] Schwartz, H.R. et al., "Single event upset in flash memories", IEEE Trans. on Nucl.Sci., vol. 44, no. 6, pp 2315-2324, December, 1997

[14] R.D. Roth et al., "SEU and TID testing of the Samsung 128 Mbit and the Toshiba 256 Mbit flash memory,"IEEE Radiation Effects Data Workshop, pp. 96-99, July 2000

[15] Nguyen, D.N and Scheick, L.Z., "SEE and TID of emerging non-volatile memories," IEEE Radiation Effects Data Workshop, pp. 62-66, 2002

[16] Miyahira, T. and Swift, G., "Evaluation of radiation effect in flash memories," Proceedings MAPLD Conference, 1998

[17] Wakerly, J.F., "Digital design principles & practices," 3rd Edition, Prentice Hall, 2001

[18] Label, K. et al., "Radiation effect characterization and test methods of single chip and multi chip stacked 16Mbit DRAMs," IEEE Trans. on Nucl. Sci. vol 43, no. 6, pp. 2974-2981, December 1996

[19] Label, K. et al., "Anatomy of an in-flight anomaly: investigation of proton-induced SEE test results for the stacked IBM DRAMs," IEEE Trans. on Nucl. Sci., vol. 45, no. 6, pp. 2898-2903, December 1998

[20] Makihara, A. et al., "Analysis of single ion multiple bit upset in high density DRAMs," IEEE Trans. on Nucl. Sci., vol. 47, no. 6, pp. 2400-2404, December 2000

[21] J. Rodgers, R. Brown and Nadim Haddad, "Advanced memories for space applications", IEEE Microelectronics Reliability & Qualification Workshop, December 2001

[22] P. Layton, "SEE radiation test report," Technical Report, Document No. 1003857, Maxwell Technologies, April 2003

[23] R. Koga, P.Yu, K.B. Crawford, S.H. Crain, V.T. Tran, "Permanent single event functional interrupts in 128- and 256- Megabit SDRAMs", IEEE NSREC Data Workshop, pp. 6-13, 2001

[24] S.M. Guertin, J.D. Patterson and D.N. Nguyen, "Dynamic SDRAM SEFI detection and recovery test results", IEEE Radiation Effects Data Workshop, pp. 62-67, 2004

[25] J. Gaisler, "Suitability of reprogrammable FPGAs in space applications,"ESA Feasibility Report, FPGA-002-01, September 2002

[26] C. Carmichael, et al., "Proton testing of SEU mitigation methods for the Virtex FPGA," Proceedings MAPLD Conference, 2001

[27] J.J.Wang et al., "SRAM based reprogrammable FPGA for space applications," IEEE Trans. on Nucl. Sci., vol. 46, no. 6, pp. 1728-1735, December 1999

[28] S. Mattsson, "Single event upset tests of commercial FPGA for space applications," Workshop on Electronics for LHC Experiments, Stockholm, Sweden, 10-14 September 2001

[27] Parkes, S., "SpaceWire: the standard," Proceedings of the Data Systems in Aerospace, pp. 111-116, 1999

[28] S.N.Chau, ;J. Smith, T. Tai, "A design-diversity based fault-tolerant COTS avionics bus network," Proceedings Of IEEE Pacific Rim International Symposium on Dependable Computing, pp. 35-42, December 2001

[29] S. Buchner, "Evaluation of commercial communication network protocols for space applications", Presented at SEEWG, Los Angeles November 2003 http://radhome.gsfc.nasa.gov/radhome/papers/SEEWG03_Network.pdf

[30] C. Seidleck et al, "Test methodology for characrterizing the SEE response of a commercial IEEE 1394 serial bus (firewire)", IEEE Trans. On Nucl. Sci, vol. 49, no. 6, 2002

[31] S. Buchner and E. Rodrigurz, "Pulsed laser test of Atmel chip", http://nepp.nasa.gov/DocUploads/C4A1CBBE-8DAF-4845-943B1E851ADAFF30/NRL_112103_ATMEL.pdf, accessed on 15-03-05

[32] S.N. Chau, L. Alkalai, A.T. Tai and J.B. Burt, "Design of a fault-tolerant COTS-based bus architecture", IEEE Trans. On Nucl. Sci. vol. 48, no. 4, 1999

[33] Parkes, S., "SpaceWire: the standard," Proceedings of the Data Systems in Aerospace, pp. 111-116, 1999

[34] S.N.Chau, ;J. Smith, T. Tai, "A design-diversity based fault-tolerant COTS avionics bus network," Proceedings Of IEEE Pacific Rim International Symposium on Dependable Computing, pp. 35-42, December 2001

[35] S. Fischer,S. Parkes. and G. Kempf , "Spacewire router", ESA Microelectronics Presentation Days, Noordwijk, Holland, 4-5 February 2004

[36] Actel Application Note, "Design techniques for radiation hardened FPGAs", 1997

[37] D. Patel Et. al, "Space qualified radiation-hardened FPGAs, A successful collaboration continues", The MAPLD Conference 2004 http://klabs.org/mapld04/presentations/session_p/p165_patel_s.pdf

[38] Actel Application Note, "RTAX-S radiation tolerant feature and mitigation techniques", 2005

[39] L. Larcher, G. Cellere, A. Paccagnella, A. Chimenton, A. Candelori and A. Modelli, "Data retention after heavy ion exposure of floating gate memories: analysis and simulation" IEEE Trans. On Nucl. Sci., vol.50, no. 6, pp. 2176-2183, December 2003

[40] ESA SpaceWire Standard, http://www.estec.esa.nl/tech/spacewire/overview/

[41] C.I Underwood, D.J. Brock, P.S.Williams, S. Kim, R. Dilao, P.R. Santos, M.C. Brito, C.S. Dyer, and A.J. Sims (1994) "Radiation environment measurements with the cosmic-ray experiments on-board the KITSAT-1 and PoSAT-1 micro-satellites". IEEE Trans. on Nucl. Sci., vol. 41, no. 6, pp. 2353-2360, December 1994

[42] C.I. Underwood, A. da Silva Curiel, and M.N. Sweeting, (2002) "In-orbt monitoring of 'Space Weather' and its effects on commercial-off-the-shelf (COTS) electronics - a decade of research using micro-satellites". Paper presented at the 53rd International Astronautical Congress, October 10th – 19th 2002, Houston. USA. (IAC-02-IAA.6.3.04).