Reconfigurable Processing Module

Kevin Somervill, Dr. Robert Hodson
NASA Langley Research Center, Hampton, VA
Dr. Robert Jones
ASRC Aerospace Corp., Greenbelt, MD
Dr. John Williams
The University of Queensland, Brisbane, AU

Military and Aerospace Applications of Programmable Devices and Technologies
International Conference
September 7–9, 2005

Abstract

To accommodate a wide spectrum of applications and technologies, NASA's Exploration System's Missions Directorate has called for reconfigurable and modular technologies to support future missions to the moon and Mars. In response, Langley Research Center is leading a program entitled Reconfigurable Scaleable Computing (RSC) that is centered on the development of FPGA-based computing resources in a stackable form factor. This paper details the architecture and implementation of the Reconfigurable Processing Module (RPM), which is the key element of the RSC system.

The RPM is an FPGA-based, space-qualified printed circuit assembly leveraging terrestrial/commercial design standards into the space applications domain. The form factor is similar to, and backwards compatible with, the PCI-104 standard utilizing only the PCI interface. The size is expanded to accommodate the required functionality while still better than 30% smaller than a 3U CompactPCI™card and without the overhead of the backplane. The architecture is built around two FPGA devices, one hosting PCI and memory interfaces, and another hosting mission application resources; both of which are connected with a high-speed data bus. The PCI interface FPGA provides access via the PCI bus to onboard SDRAM, flash PROM, and the application resources; both configuration management as well as runtime interaction. The reconfigurable FPGA, referred to as the Application FPGA – or simply "the application" – is a radiation-tolerant Xilinx Virtex-4 FX60 hosting custom application specific logic or soft microprocessor IP. The RPM implements various SEE mitigation techniques including TMR, EDAC, and configuration scrubbing of the reconfigurable FPGA. Prototype hardware and formal modeling techniques are used to explore the performability trade space. These models provide a novel way to calculate quality-of-service performance measures while simultaneously considering fault-related behavior due to SEE soft errors.

1 Introduction

Advances in computing platform requirements have increased in all venues of digital systems design including those targetted for space-based applications. Coupled with this is the desire to develop new systems at increasing rates without loss of confidence in the reliability or performance of the design. To meet this challenge, NASA Langley Research Center (LaRC) is leading the Reconfigurable Scalable Computing (RSC) project to develop and deliver a modular computing platform that is both scalable and reconfigurable. Led by Dr. Robert Hodson, the RSC project is developing several modules over the course of four years that

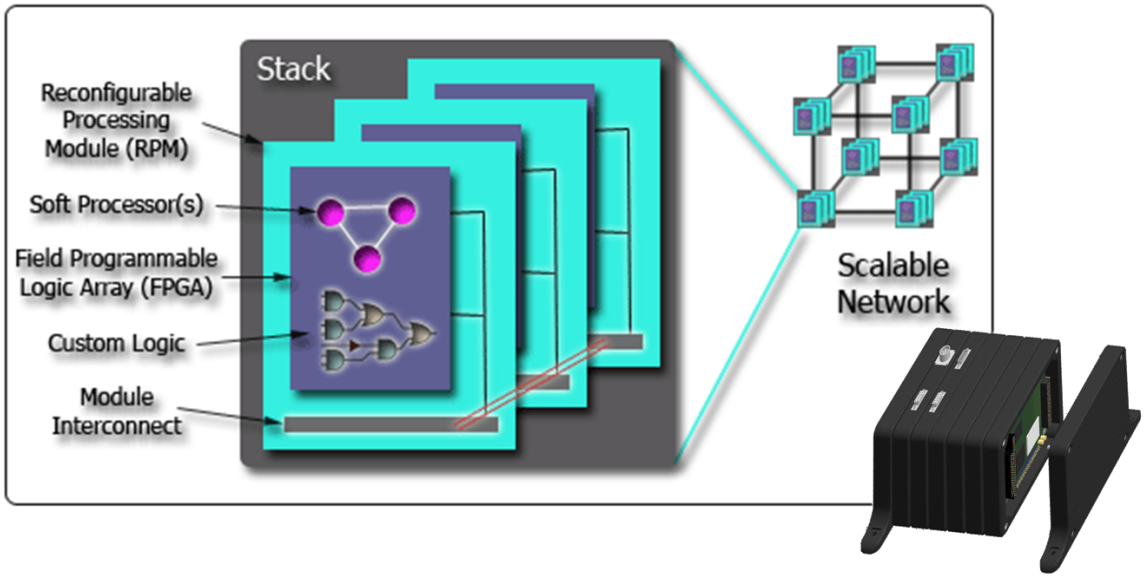

Figure 1: RSC platform architecture

are PCI-104 "bus compatible" [1][1] and stack together providing modular designs that are both scalable by adding more modules to a system and reconfigurable by interchanging modules to accommodate different mission requirements. A second dimension to the reconfigurable aspect of an RSC platform is the use of reconfigurable SRAM-based Field Programmable Gate Array (FPGA) devices on what is referred to as the Reconfigurable Processing Module (RPM).

An example RSC system is illustrated in Figure 1 with a rendering of an RSC stack in the lower right corner. The figure shows the logical and physical connections present in a typical RSC system. Though only the RPM is named in the figure, two or more modules can be combined to form a stack, and multiple stacks can be connected to develop a network system.

As stated above, the reconfigurable aspect of the RPM is central to RSC as a platform. The following sections describe the characteristics and development of the RPM including major components, fault-tolerance techniques and development methodologies implemented to ensure a robust space-worthy design. The discussion starts with a presentation of the architecture in section 2 followed by a description of the form factor presented in section 3. Fault mitigation is covered in section 4 and then section 5 presents aspects of the development process and software support. Section 6 presents a few example applications. Finally, the paper concludes with a few closing remarks in section 7.

2 Architecture

The RPM architecture is not unlike typical computing systems with a processing element, a memory subsystem, input/output (I/O) interfaces, and a communications bus. The key divergence from the typical is that the processing element is hosted in reconfigurable FPGA fabric[2] in what is referred to as the Application

[1] PCI-104 "bus-compatible" means that the PCI interface is mechanically and electrically compatible with standard PCI-104 products, but may have different physical dimensions.

[2] Fabric is common terminology for interconnect logic. In the case of an FPGA, the "fabric" is composed of logic cells and distrubted memory. Another example at a larger scale would be a network of computing elements where the I/O interconnect serves as the fabric.

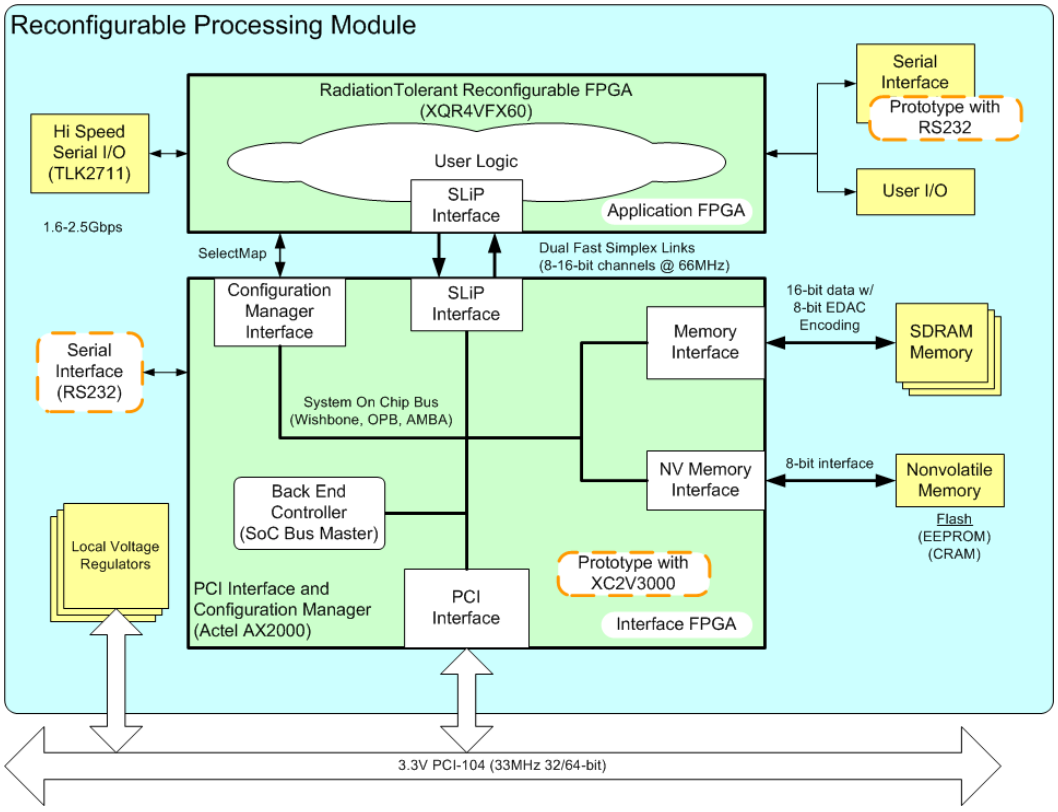

Figure 2: RPM toplevel architecture

FPGA in the upper portion of Figure 2. As illustrated, the Application FPGA is supported by what is referred to as the Interface FPGA shown in the lower half of the figure, a memory subsystem including Synchronous Dynamic RAM (SDRAM) and non-volatile memory, a Peripheral Component Interconnect (PCI) Local Bus interface implemented in the Interface FPGA, local DC regulation, a high speed serial interface, and user definable I/O.

2.1 The Application FPGA

The Application FPGA is a Xilinx Virtex-4 FX60 that hosts two primary interfaces, one connected to a Texas Instruments TLK2711 Mutlit-Gigabit Transceiver providing a 2.5Gbps serial interface, and one to the Interface FPGA. Also connected to the Application FPGA is a UART providing a slow serial interface (RS422 in a flight configuration and RS232 for the prototype as a development convenience) and user definable I/O. The UART is externally accessible via a 25-pin D-type connector that also hosts a few discrete I/O. The majority of user I/O is routed through one of the stackable connectors as an auxillary bus next to the PCI bus. To facilitate communications between the hosted application and the rest of the system, a pair of simplex busses (referred to as the Simplex Link Pair, or SLiP) provide fully concurrent interfaces between the Application FPGA and memory, networking, or the PCI bus. The SLiP bus is comprised two 16-bit unidirectional busses (channels) operating at 66MHz to 100MHz yielding approximately 200MB/s per channel, or an aggregate of approximately 400MB/s total bandwidth through the SLiP interface. A number of control signals, in addition to the sixteen data signals, provide control flow and status for the busses. Cores for each of the interaces will be provided as part of the development environment.

The TLK2711 transceiver provides a high speed interface directly into the Application FPGA, providing a mechanism to process incoming data without having to store it in memory first. The interface to the FPGA is two 16-bit unidirectional busses (a receive side and a transmit side) that operate in the range of 80MHz to 125MHz providing 1.6Gbps to 2.5Gbps data rate on the external side of the transceiver. The TLK2711 device performs 8b/10b encoding and decoding, clock embedding and extraction to and from the serial stream and is low power at less than 500mW. The prototype will use commercial components and a flight configuration will use the hermetically sealed ceramic package available early next year.

2.2 The Interface FPGA

The interface FPGA serves to provide a stable environment for the application. As such, it is not reconfigurable after deployment and serves to host the PCI interface providing access to the rest of the stack or network of stacks. As shown in Figure 3, the aptly named Interface FPGA provides all the necessary interfaces between the RSC platform and local resources. For the prototype, the Interface FPGA is implemented in a Xilinx XC2V3000-BG728 to enable testing of various architectural implementations with real hardware. For a flight configuration, an Actel RTAX2000S-1CG624E will be used to host the Interface FPGA implementation and though it is not flight reconfigurable, being an FPGA still allows for custom tweaks to tune the Interface FPGA functionality to specific environmental and project requirements.

As can be seen in Figure 3, interfaces to volatile and non-volatile memory, PCI, and the Configuration Manager (CM) for the Application FPGA are interconnected via an on-chip bus. This on-chip bus is tentatively planned to be implemented using the WISHBONE interface [2], but could readily be supported using IBM's CoreConnect™ On-chip Peripheral Bus (OPB). The RPM PCI interface includes Master/Target functionality with DMA capability for 32-bit and 64-bit 33MHz PCI bus. The 33MHz 32-bit PCI bus is standard for the PCI-104 specification, and an additional stackthrough connector has been added to support the 64-bit PCI extension.

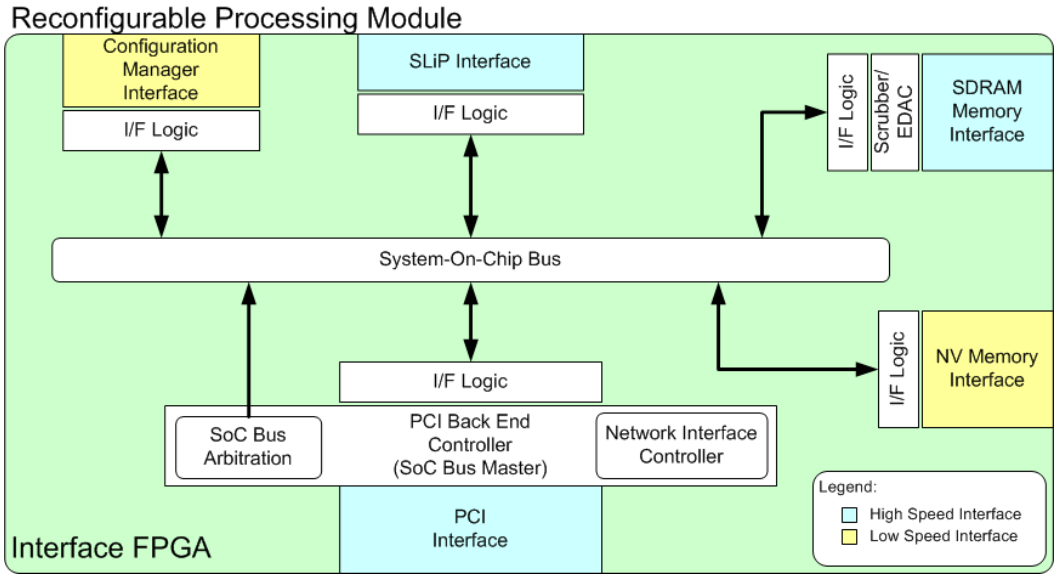

Figure 3: RPM toplevel architecture

2.3 The Memory Subsystem

The memory subsystem is comprised of three SDRAM modules and non-volatile memory storage (FLASH in the case of the development prototype). The SDRAM is the 3D-Plus 2-Gigabit modules, part number MMSD08256804S-C. These are organized as 256Mx8-bit and built from a stack of four Elpida 64Mx8 SDRAM components (part number EDS5108ABTA-75). The 3D-Plus 2Gbit SDRAM on the RPM is a 54-pin TSOP-*like* package nominally 7.45mm (0.293in) tall which is critical to the 8.76mm (0.345in) height constraint for PCI-104 compatibility [1]. The three SDRAM memory modules share an address and control bus and provide a 16-bit data interface with eight check-bits. The SDRAM operates at 66MHz or 100MHz, matching or slightly exceeding 32-bit PCI bandwidth.

The non-volatile memory is implemented in Intel StrataFLASH for the protoype design as a robust implementation with sufficient capacity has not been identified for the flight configuration. The RPM will support at a minimum 8MB[3] of non-volatile memory storage. Possible flight implementation alternatives include screened commercial components that are powered down when not in use or BAE's Chalcogenide RAM (CRAM) currently under development. Should it be determined that a sufficient capacity non-volatile memory solution simply will not fit on the RPM, another board will have to be developed to provide this functionality for a stack, or system.

3 Physical Description

Though each RSC module, including the RPM, has its form factor derived from the PCI-104 standard, designing for the space environment requires catering to larger component sizes and thermal management issues not present in commercial systems. To meet these requirements, the base specification has been augmented with consideration for larger printed circuit board (PCB) area and conduction cooling. Previously, NASA has adapted PC/104-*Plus* architecture for space [3] making a number of changes in form factor

[3] A note on nomenclature, MB refers to MegaBytes while Megabits would be denoted as Mbits.

and connectivity. RSC, in a similar manner has increased the size from 90.17mm (3.550in) x 95.89mm (3.775in) by 1.962in on one side to a dimension of 140mm (5.512in) by 95.89mm (3.775in) as shown in Figure 4, but has maintained connector compatibility. Instead of adhering to the older PC/104 or PC/104-*Plus* standards, the PCI-104 base standard has been selected as it does not include the lower speed ISA bus. The area for the ISA connector is used for component placement. RSC modules are PCI-104 compatible and can mate with *Commercial-off-the-shelf* (COTS) PCI-104 hardware. The 120-pin 2mm PCI connector is unchanged from the standard, but a second 120-pin 2mm connector has been added for the possibility of supporting the PCI 64-bit extension as a means to increase PCI bus bandwidth. The PCI operates at 33MHz and is connected to any card within a stack as it supplies the system interface for data transport and power. Based on preliminary estimates, an active RPM is expected to consume a nominal 10W and a maximum of approximately 18W. Since convection cooling does not work in the absence of gravity and PCI-104 contains no remarks with respect to conduction cooling, other accommodations had to be made to account for thermal management for the RPM and other RSC modules. To account for this, the VITA 30.1-2002 standard for conduction cooling is used as the prescribing specification, similar to CompactPCI™, and thermal bands[4] outside the standard PCB area have been added as a thermal conduction path to the carrier (frame) and then to the mounting plate. The arrangement of the RPM PCB in the carrier is shown in Figure 5. The carrier has a ledge for the thermal interface to the PCB. Thermal conductivity is ensured with a compression gasket that overlays the perimeter of the PCB securing it to the carrier. In a completely assembled RSC stack, the carriers interconnect to form the system enclosure and provide Electro-Magnetic Interference (EMI) and radiation shielding. All of these deviations from the PCI-104 standard are being captured by NASA Langley personnel in an addendum to the PCI-104 specification referred to as Space-104. The development of this addendum is open to interested parties.

As can be seen in Figure 4, the RPM is quite dense. The two largest devices are FPGA devices for the Interface FPGA (left) and the Application FPGA (right). The two 120-pin connectors found along the top left edge and far left are the standard 32-bit PCI and PCI 64-bit extension connectors, respectively. Continuing around the edge of the PCB, the three 54-pin components at the bottom left of the PCB are three SDRAM modules providing 512MB of memory for the RPM. To the right along the bottom edge there is a 25-pin D-type connector used for external I/O. At the far right edge there are four impedance controlled connectors used for high speed serial I/O. Also present on the prototype shown in Figure 4 are voltage regulators, debug connectors, and flash memory.

4 Mitigation

There are a number of mitigation methods utilized in the design of the RPM to handle Single-Event Upsets (SEU). Among these are careful part selection, Triple Modular Redundancy (TMR), Error Detection and Correction (EDAC), and data scrubbing. All of these are implemented to protect a specific aspects of the design from data integrity to actual hardware configuration and are briefly described here.

Paramount to each of these mitigation methods is the selection of qualified radiation-tolerant parts. For the RPM, the intent is to utilize components that can endure 100krads Total Ionizing Dose (TID) and are latch-up immune. This has proven difficult for non-volatile memory of any significant capacity, SRAM-based FPGA devices like the Application FPGA, and low voltage DC regulation. The non-volatile memory is a currently unresolved industry-wide issue with mitigation methods including latch-up detection circuitry and removing power from the device to allow for annealing of the transistors, both of which are less than ideal. TMR methods are applied to the Application FPGA as described next. Low voltage DC regulation is another industry-wide dilemma gaining attention as core voltages for devices drop below 2V. One approach is to utilize linear regulation (currently with discrete components), but this only worsens the thermal management as the regulation from say 3.3V to 1.2V requires considerable power dissipation. Another alternative is the

[4]The thermal bands are not shown in Figure 4.

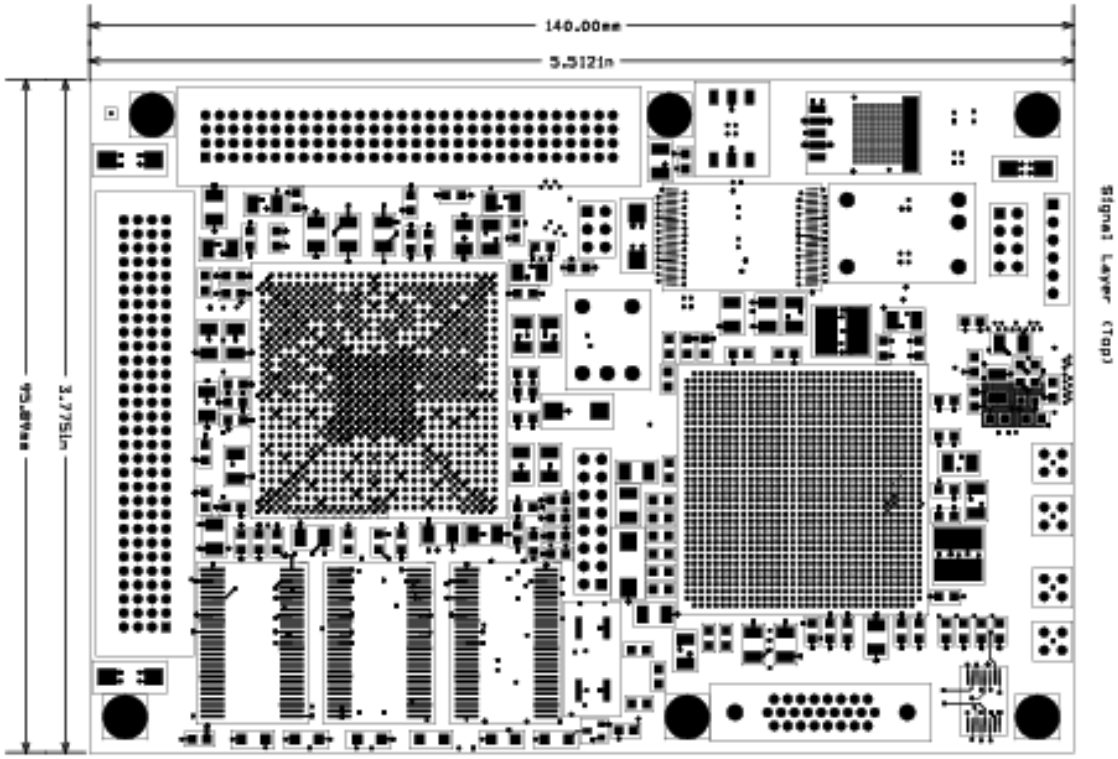

Figure 4: Top layer of RPM PCB.

implementation of switching DC regulators providing much higher efficiency than their linear counterparts. These are the current leading challenges in developing a completely radiation-tolerant bill of materials for the RPM and other RSC modules.

To mitigate SEUs in the large FPGA devices, TMR is implemented in different forms in both the Application and the Interface FPGAs. For the Application FPGA, the Xilinx XTMR tool [4] is used to triplicate logic and I/O as well as filter out susceptible structures like SRL16s and LUT_RAMs replacing them with flip-flops. The North American Single Event Effects Consortium has performed a number of tests on SRAM-based FPGA devices with results that indicate that robust designs for custom logic and soft processor cores [5] are possible with the appropriate application of mitigation techniques. The Interface FPGA, an RTAX2000 provides inherent TMR at the device level[5].

To protect the integrity of the data, scrubbing is performed in several portions of the RPM design. Due to the well-known SEU susceptibility of the configuration registers of SRAM-based FPGAs (including the Application FPGA), a Configuration Manager (CM) is implemented to configure and maintain the device programming of the application. The CM serves to maintain a known functional configuration by continuously updating the configuration of the Application FPGA. This functionality is referred to as "scrubbing" and the process occurs concurrently with the application operating uninterrupted. Configuration of the Application FPGA is controlled through the PCI interface by the system controller which will load configuration from lo-

[5]Since the prototype uses a Xiliinx XC2V3000 for the Interface FPGA, the device is not TMR, but this should be sufficient and nearly equivalent for the purposes of demonstrating the platform

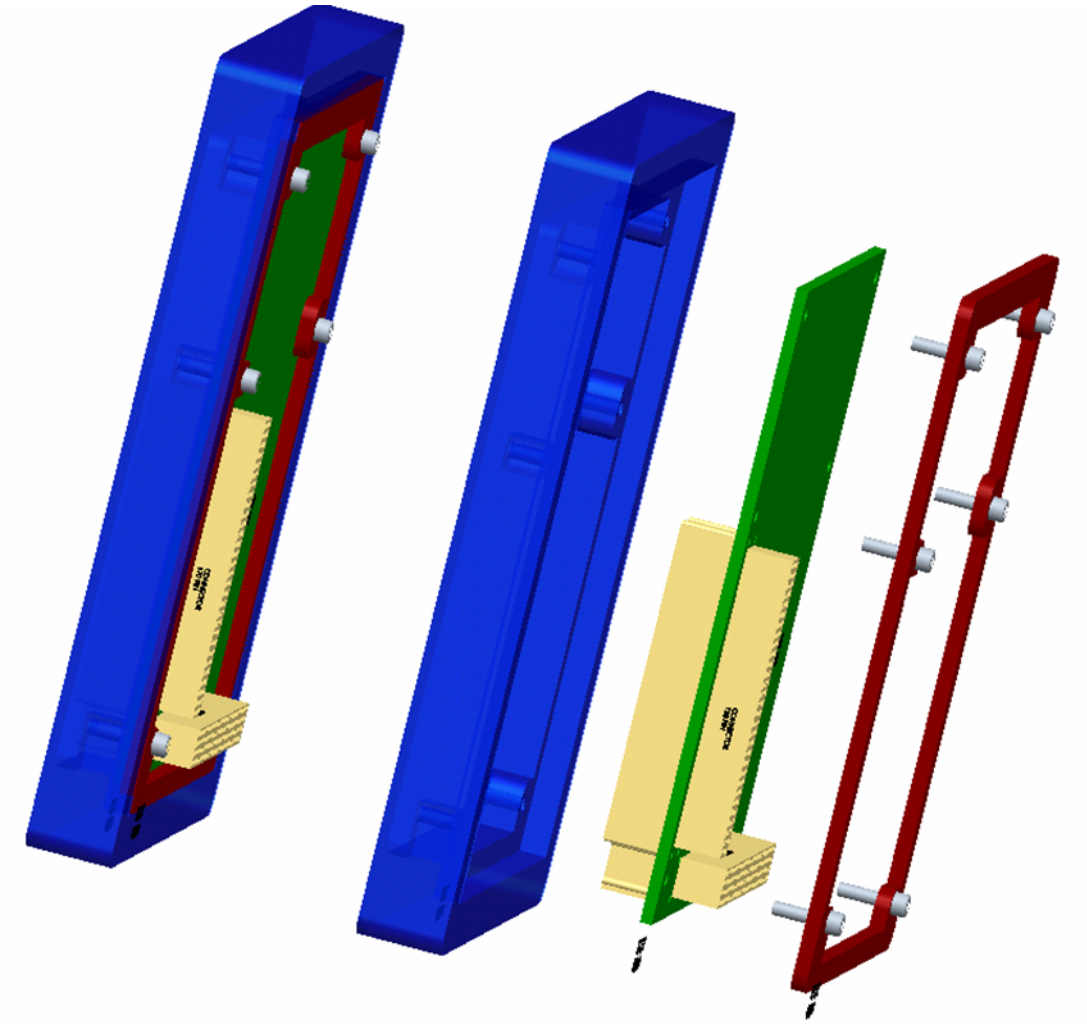

Figure 5: Assembly of the RPM and the carrier frame.

cal flash to a location in the SDRAM, or load a new image directly to SDRAM from another source (ground operations for example). Upon configuration of the application, the CM will continuously reprograms the application to ensure correct operation of the design. Another form of scrubbing implemented in the RPM is applied to the SDRAM for the purpose of maintaining data integrity and preventing the accumulation of upsets in the memory store. For this process, data is periodically read from the SDRAM and any corrected values are written back to memory.

To locate and correct errors during the scrubbing process, EDAC is implemented. On initial boot of the system, data is be loaded from the non-volatile store to SDRAM and checked with block level encoding, like Reed-Solomon (RS) for example. The actual RS decoding will be performed by the RSC control module and not performed by hardware in the Interface FPGA due to the size of RS decoder cores and the infrequent nature of accessing the non-volatile memory. For the SDRAM, a hardware EDAC implementation will provide, at a minimum, Single Error Correction Double Error Detection (SECDED). With a 24-bit data interface to the three SDRAM modules, 16-bit data is protected with 8 check-bits including parity. Currently, Double Error Correction Triple Error Detection (DECTED) is being implemented as the EDAC engine embedded

into the SDRAM interface. It is believed that this level of error correction will be appealing to applications targetted for harsher environments.

5 System Modeling and Software

5.1 Development Process

A model-centric approach is being applied to system development which also contributes to risk mitigation. As Figure 6 shows, this approach enjoys the benefits of three novelties in all phases of development: 1) abstraction using one or more modeling formalisms, which can be appropriately chosen and mixed together in a single model study, 2) model checking via exact state-space exploration, and 3) model analysis via performability measures. Our goal is to maximize system *performability* – the unified assessment of system performance and its dependability of service within the same model – while minimizing risk by appropriately coupling standard practices with modern advancements in more formal (model-driven) design and development techniques. To this end, abstract models are used to capture and analyze the system throughout its development from architectural concepts to design details. With sufficient expressive power and convenience of abstraction, model construction can mature through iterative refinement. Models can thus drive the development of hardware and software while always ensuring unambiguous specifications and continuity from one phase of development to the next.

Key to the RSC development process is the *framework of modeling paradigms* that vary in modeling power, efficiency, and convenience. The modeling formalisms we use include Petri nets [6], SystemC [7], and VHDL, which can range in levels of abstraction from process flow and transaction levels down to the register transfer level (RTL). The Petri net (PN) has the advantage of being both mathematically and graphically based, and it can more easily abstract away details at arbitrary levels. So, as a well-defined mathematical formalism, PN models are suitable for direct input into a computer for analysis at any phase of development. In graphical form, PN models are also suitable for human consumption to aid the expression of ideas, the specification of designs for review, or complement the technical documentation. We therefore take advantage of the modeling power and convenience of the PN language to drive the development of SystemC and VHDL models throughout system development. More specifically, PNs can capture models ranging from process flows, data and control flow, queuing networks, and finite automata, to arbitrary algorithms including Turing-complete computation. SystemC and VHDL models, in turn, can facilitate the interfacing and co-simulation of hardware and software components that realize the algorithms, as well as the RTL specification and clock-accurate simulation of the FPGA design details that are synthesized into the system.

At the heart of this model-centric framework is an innovative software tool. Referred to as the Stochastic Model-Checking Analyzer for Reliability and Timing, or SMART [8], it is a tool with roots in academia, and its development funded in part by the NASA Langley Research Center. SMART provides the most advanced toolset for the logic and probabilistic analysis of complex systems. Models can be constructed using a variety of formalisms including PNs and Markov chains (probabilistic state machines). Custom formalisms can also be incorporated by specifying the language grammars. Both explicit and symbolic state-space generation techniques, as well as symbolic computation tree logic (CTL) algorithms, are also available [9]. All this allows extremely large state spaces to be generated and explored on a modern desktop computer.

Finally, as part of the modeling-and-development process flow, a SystemC model will be constructed for use as the VHDL testbench. The efficacy of this testbench generation will then be evaluated as a new "best practice" in digital system design. While we expect that this additional task in the process flow will increase the development time of the RPM in particular – being the first module to be developed – we also anticipate significant savings in time later when designing each subsequent module within the RSC suite since the testbench will be reusable.

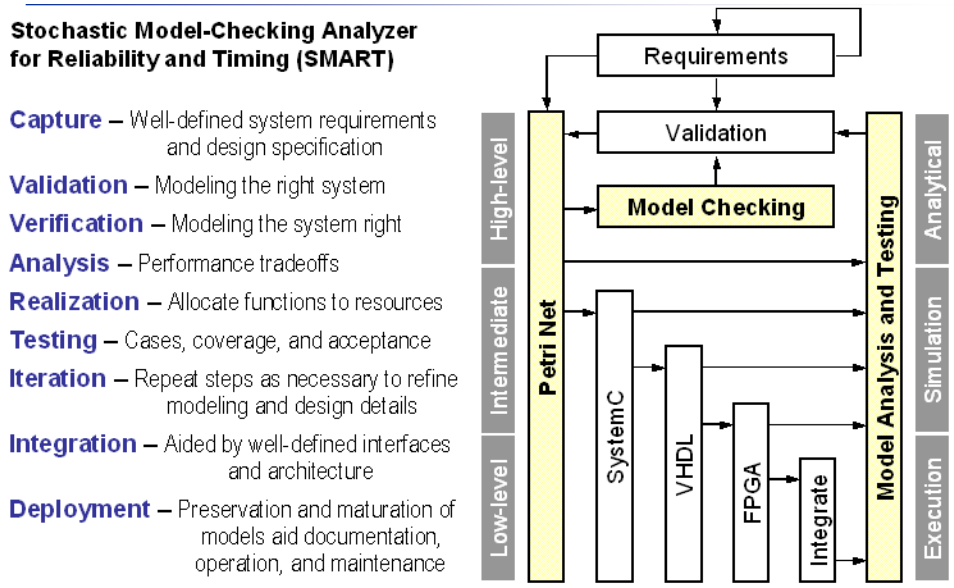

Figure 6: "SMART approach" to model-centric engineering of the RPM and RSC module suite.

5.2 μClinux and the Message Passing Interface

As part of the software development, the RSC will be leveraging research efforts on the support for operating systems on soft processor cores. Specifically, the RPM will support μClinux on the MicroBlaze™ processors providing a development environment not unlike that of a typical Linux desktop. While the MicroBlaze™ does not have a Memory Management Unit (MMU), μClinux is a port of Linux specifically addressing the requirements of such processors. The μClinux operating system boasts a small memory footprint and almost all of the conveniences of main stream Linux. Partners in the RSC project have previously ported μClinux to the MicroBlaze™ processor demonstrating it to be an effective operating system, not only in embedded applications, but also reconfigurable systems [10]. This work directly conveys to applications in parallel processing with embedded soft processors. As part of parallel processing, a subset of the Message Passing Interface (MPI) library is being ported to μClinux hosted on an RPM. This will allow a network of RSC stacks to function, to the extent possible, like a terrestrial Linux cluster. Initially, this will be software-only, but as the project progresses various functionality will be migrated to VHDL logic cores utilizing the FPGA fabric for Linux / custom HW integration exposing the capabilities of the FPGA logic. With support for Linux with its inherent tools and capabilities coupled with the MPI library, a rich development environment is made available for parallel applications development for space targetted platforms.

6 Example Applications

With the completion of the description of the RPM, it seems appropriate to detail a few potential architectures. The following subsections present high level overviews of systems similar to planned demonstration platforms. The first includes an example of a soft processor based application. The second describes an application implementing custom logic.

6.1 Soft Processor

As part of the project level objectives, the RPM will support soft core processors, specifically the MicroBlaze™, to enable parallel processing for space applications as well as support standard software development for applications written in C. Consider an imaging system with a front-end instrument routing data to an RSC stack. The data is packetized and sent out the high speed serial interface to a remote (but perhaps identically configured) RPM for processing. Such a configuration for the RPMs may resemble that shown in Figure 7. This Application FPGA hosts two MicroBlaze™ processors, one minimally configured for I/O packet manipulation (parsing datagram headers, etc) and another fully configured to host μClinux as a local controller and data processing engine. As shown, the two processors could communicate via OPB bridge, or Fast Simplex Links (FSL), and would share the resources provided by the Interface FPGA. Since the MicroBlaze™ does not currently have an MMU as a configuration option, true Symmetric Multi-Processing (SMP) functionality is problematic, but relegating one processor to the function of a co-processor minimizes competition for SLiP interface and memory resources. As the project progresses, research will proceed on implementing true SMP in a single RPM.

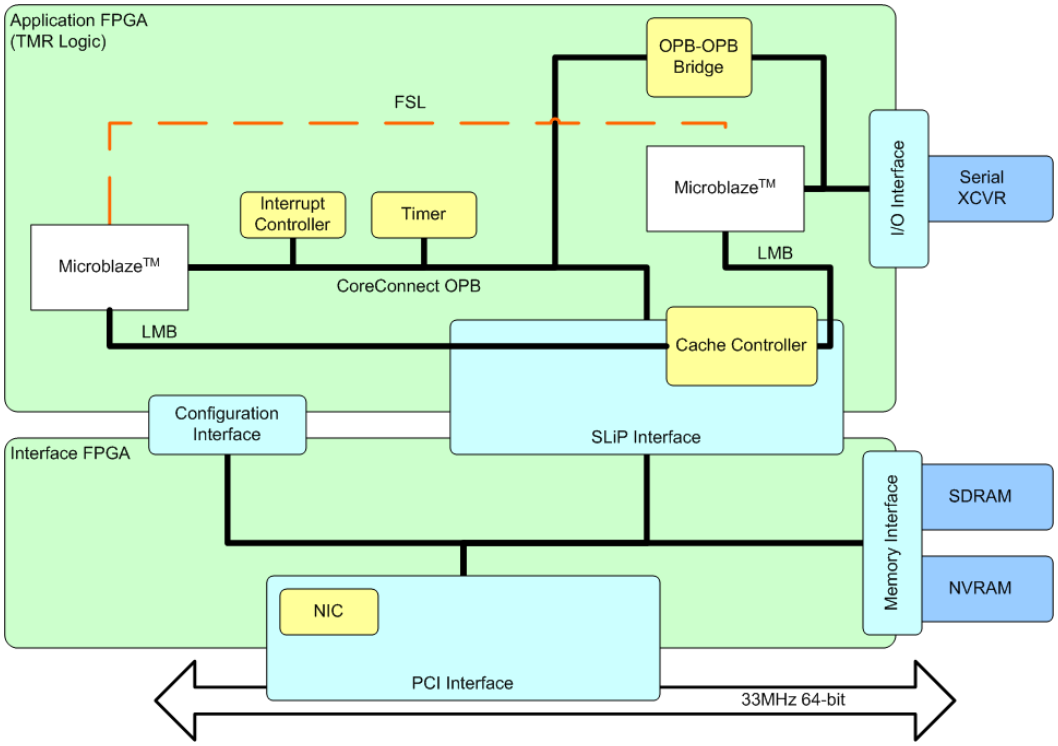

Figure 7: Example soft processor application implementation.

6.2 Custom Logic

Figure 8 illustrates a potential architecture for an image processing application utilizing custom logic cores. In this example, input data is again received from the high speed serial interface, this time routed to an Image Processing Engine that may implement an object tracking algorithm for example. Data is then forwarded to filtering logic that would feed a fuzzy logic controller. Such a configuration may be useful for

an avoidance and tracking control system in an autonomous rover where the fuzzy logic controller would provide control data to a drive chain control system.

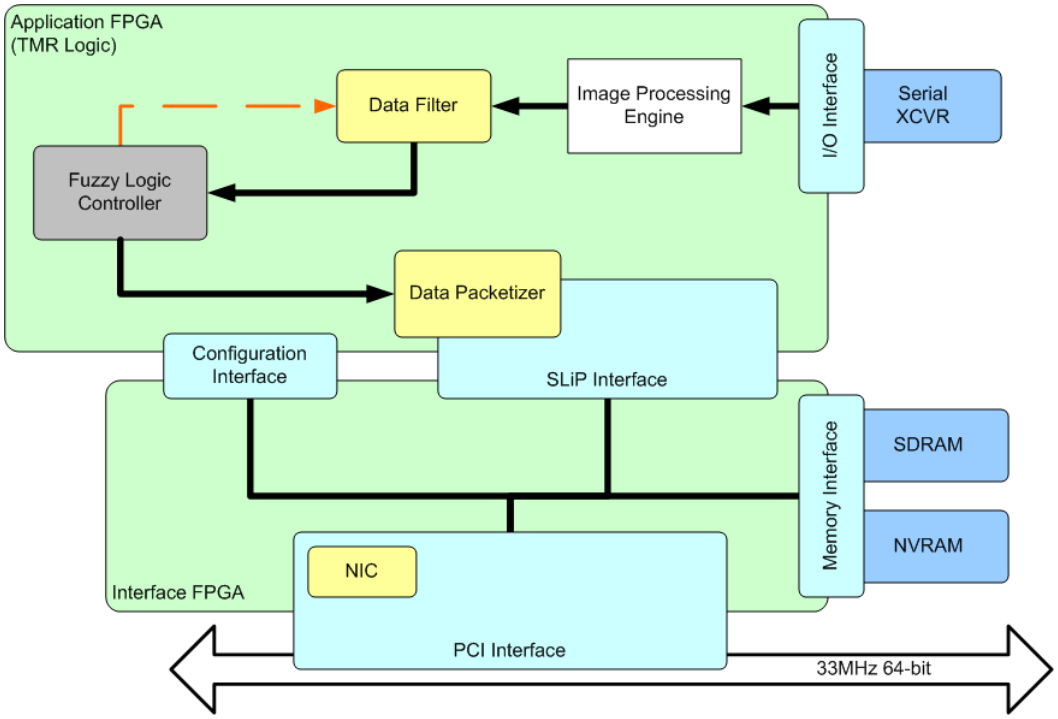

Figure 8: Example custom core application implementation.

7 Conclusion

The Reconfigurable Processing Module has been presented detailing the hardware description and functionality. The overall architecture has been introduced describing the key resources available to hosted applications. A brief discussion of the form factor has highlighted a few enhancements made to the PCI-104 specification to enable the stackable architecture for space flight operation. Fault mitigation techniques and design methodologies have been presented illustrating the considerable effort going into ensuring a robust radiation-tolerant platform. Software and operating system support have been presented and provide a familiar and robust computational platform for a myriad of applications. Two example applications have been presented that illustrate potential platforms for RSC systems as well as their efficacy in those platforms.

The development of the RPM as part of the Reconfigurable Scalable Computing project serves to demonstrate the superiority of reconfigurable architectures over traditional processor architectures in applications that perform highly regular computations. To accomodate tasks with highly diverse functionality, where microprocessors use logic resources more efficiently than reconfigurable devices, soft processor cores are imlemented. With the approach of developing a mixed architecture of special-purpose cores of computing functions and soft-processor cores, each reconfigurable, an efficient solution for a given computational task can be found by exploiting the strengths of both architectures.

References

[1] PC/104 Embedded Consortium, *PCI-104 Specification*, November 2003. Revision 1.0.

[2] OpenCore.org, *WISHBONE System-on-Chip (SoC) Interconnection Architecture for Portable IP Cores*, September 2002. Revision B.3.

[3] D. L. Abbott, G. Cox, and H. Nguyen, "Adapting PC/104 for space," *PC/104 Embedded Solutions*, 2000.

[4] C. Carmichael, "Triple Modular Redundancy Design Techniques for Virtex FPGAs," Application Note 197, November 2001.

[5] S. Rezgui, G. Swift, K. Somervill, J. George, C. Carmichael, and G. Allen, "Complex Upset Mitigation Applied to a Reconfigurable Embedded Processor," in *IEEE Nuclear and Space Radiation Effects Conference (NSREC'05)*, (Seattle, Washington), July 2005.

[6] T. Murata, "Petri Nets: Properties, Analysis, and Applications," in *Proceedings of the IEEE*, vol. 77, pp. 541–580, April 1989.

[7] Open SystemC Initiative, San Jose, CA 95118-3799, *SystemC 2.0.1 Language Reference Manual*, 2003. Revision 1.0.

[8] G. Ciardo, R. L. Jones, III, R. M. Marmorstein, A. S. Miner, and R. Siminiceaunu, "SMART: Stochastic Model-checking Analyzer for Reliability and Timing," in *Proceedings Int. Conference on Dependable Systems & Networks (DSN)*, (Washington, D.C.), June 2003.

[9] G. Ciardo, R. L. Jones, III, A. S. Miner, and R. Siminiceaunu, "Logical and Stochastic Modeling with SMART," in *Proceedings of the 13th Int. Conf. on Modelling Techniques and Tools for Computer Performance Evaluation*, (Urbana-Champaign, IL), pp. 78–97, September 2–5 2003.

[10] J. A. Williams and N. W. Bergman, "Embedded Linux as a Platform for Dynamically Self-Reconfiguring Systems-On-Chip," in *In Engineering of Reconfigurable Systems and Algorithms (ERSA 2004)*, (Las Vegas, Nevada, USA), June 21–24 2004.