RECONFIGURABLE FAULT TOLERANCE: A FRAMEWORK FOR ENVIRONMENTALLY ADAPTIVE FAULT MITIGATION IN SPACE

Adam Jacobs, Alan D. George, and Grzegorz Cieslewski
NSF Center for High-Performance Reconfigurable Computing (CHREC)
Department of Electrical and Computer Engineering, University of Florida
email: {jacobs, george, cieslewski}@chrec.org

ABSTRACT

Commercial SRAM-based FPGAs have the potential to provide aerospace applications with the necessary performance to meet next-generation mission requirements. However, the susceptibility of these devices to radiation in the form of single-event upsets is a significant drawback. TMR techniques are traditionally used to mitigate these effects, but with an overwhelming amount of extra area and power. We propose a framework for reconfigurable fault tolerance which enables systems engineers to dynamically change the amount of redundancy and fault mitigation that is used in an FPGA design. This approach leverages the reconfigurable nature of the FPGA to allow significant processing to be performed safely and reliably when environmental factors permit. Phased-mission Markov modeling is used to estimate performability gains that can be achieved using the framework for two case-study orbits.

1. INTRODUCTION

As space-based remote sensor technologies increase in fidelity, the amount of data being collected by orbiting satellites and other space vehicles will continue to outpace the ability to transmit that data down to ground stations. This downlink bottleneck, caused by bandwidth limitations and high latency, can be alleviated by performing far more data processing on-board, which in turn requires high-performance computing in space. Even for applications that do not require a large downlink channel, improved on-board computational performance can enable more complex and autonomous capabilities. Additionally, in order to keep up with real-time constraints, the processing capabilities of future space systems must be increased substantially. Meanwhile, conventional systems for space typically feature a system-level design philosophy for the worst-case scenario, which can unnecessarily and dramatically hinder performance given that radiation hazards are often changing during the mission. If future space systems are to be able to achieve the high levels of performance expected, more sophisticated and adaptive methods may be necessary.

In addition to the performance requirements of these systems, space environments impose several additional constraints that are not always high priorities for terrestrial applications. System size, weight, and power (SWaP) all have major impact upon design decisions, set by physical limitations of the space platform, as is dissipation of heat. Traditionally, radiation-hardened devices, which are resistant to Single-Event Upsets (SEUs), are used to help ensure reliability and correctness in the harsh radiation environment of space. While radiation hardening at the device level makes components much more resilient to both transient and permanent faults, these devices are very expensive and often dramatically lag in performance compared to commercial-off-the-shelf (COTS) components.

One approach in designing high-performance systems for space computing is to feature hardware-adaptive devices such as FPGAs to achieve a high level of performance per unit size, mass, and power. By configuring an FPGA to perform various application-specific tasks, a reconfigurable computing system can achieve performance approaching that of application-specific integrated circuits (ASICs) while maintaining the flexibility of general-purpose processors. With this ability for reconfiguration, a smaller and lighter yet powerful system can be designed. Unfortunately, most radiation-hardened FPGAs are antifuse or flash-based and have reconfiguration limitations and much smaller capacities than their commercial counterparts. Instead, using COTS SRAM-based FPGAs in a space-based system may be a viable approach. The primary limitation of using SRAM-based FPGAs is the possibility of SEU faults causing errors in the FPGA configuration memory. The use of Triple-Modular Redundancy (TMR) and scrubbing techniques can significantly decrease occurrence of such errors.

Designing an FPGA-based satellite system using TMR can protect the system from most SEUs but with a penalty of at least 200% overhead. However, SEU rates vary over the course of an Earth orbit, and the majority of an orbit is spent in areas with relatively low upset rates. A system budgeted for worst-case scenarios will result in wasted processing time during these frequent, low-upset periods.

In this paper, we propose and demonstrate a framework for Reconfigurable Fault Tolerance (RFT) that enables FPGA-based systems to change at run-time the amount of fault tolerance being used and thereby adaptively reap a much higher degree of performance while maintaining reliability. A hardware framework for run-time support of multiple fault tolerance schemes on Xilinx FPGAs is

This work was supported in part by the I/UCRC Program of the National Science Foundation under Grant No. EEC-0642422.

presented, along with descriptions of several possible use scenarios. Dynamic partial reconfiguration (PR) is featured to change the amount of fault tolerance being used on application modules during run-time, with zero system downtime. This system of adaptive fault tolerance allows for an efficient use of resources while providing reliability necessary for space-based missions. Modeling and performability analysis of the proposed framework is shown in the context of two relevant case studies.

The remaining sections of this paper are organized as follows. Section 2 surveys previous work related to this topic. Section 3 gives an overview of the RFT architecture as well as details of possible different operating modes. Section 4 describes the modeling approach used to examine the effectiveness of our RFT approach. Section 5 analyzes results obtained using the RFT analytical model with two case studies. Finally, Section 6 presents conclusions and outlines directions for future research.

2. BACKGROUND

Traditional design approaches for space-based systems using reconfigurable FPGAs feature the use of spatial TMR. There are two primary TMR methods, external and internal. External TMR uses three independent FPGAs working in lockstep. Outputs from each device are sent to an external radiation-hardened voter that compares the results. This approach requires significant hardware overhead, but is considered very reliable. Internal TMR creates three identical modules within a single FPGA, and voting may occur either internally or externally [1]. This approach helps conserve the amount of hardware required, but may increase the chance of a common-mode failure. There are several tools available to help developers apply TMR strategies to their designs [2-3].

In addition to TMR, scrubbing is used to prevent the accumulation of errors in FPGA configuration memory. While TMR can mask individual errors, it does not correct the underlying fault and cannot correct errors that occur in multiple modules. Scrubbing uses an external device to read configuration memory of an FPGA and compare it to a known good copy. If a mismatch is detected, the correct configuration can be written using partial reconfiguration without the need to stop the operation of the device [4]. Certain components, such as LUT-RAM and SRL16s, cannot be scrubbed without corrupting user data.

System reliability and availability of these TMR systems can be accurately modeled using Markov models. In [5], continuous-time, discrete-state Markov models are used to predict system availability of the NASA Dependable Multiprocessor (DM) system developed by Honeywell and the University of Florida. Only errors caused by SEUs are considered; errors caused by natural hardware failure are ignored. The paper shows that the COTS-based system was able to meet availability requirements for missions in low-earth orbit. Although the DM system does not primarily feature FPGAs (such is considered an option), their modeling approach used can be applied to an FPGA-based system.

For systems whose configuration changes over time, such as a system with reconfigurable FPGAs, the system needs to be modeled as a phased-mission system. These systems are described by a set of unique models for each 'phase' of a mission. The states of a given model will be mapped onto the states of the following model during phase transitions. Phase duration can be modeled as either probabilistic or deterministic [6-7].

In addition to system availability, Markov reward models can be used to measure the *performability* of a system [8-9]. Performability combines system availability with the amount of work produced by the system to give a measure of total work performed and is especially useful for gracefully degradable systems. Assume that $X(t)$ is a semi-Markov process with state space S and is continuous over time $t \geq 0$, instantaneous performability is defined below where $Perf(a)$ is the system performance in state a. System performance can be defined using any desired performance metric (throughput, execution time, etc.); performability will be measured similarly.

$$Performability(t) = \sum_{a \in S} Perf(a) \cdot P\{X(t) = a\} \quad (1)$$

3. RECONFIGURABLE FAULT TOLERANCE

The motivation behind the RFT concept stems from the desire to feature some or all COTS components in a space computing system to achieve high performance. Traditional fault tolerance approaches outlined in the previous section provide methods for using these components reliably, while partially sacrificing the potential performance gains. Beyond traditional, spatial TMR, there are many alternative fault-mitigation strategies that can be situationally appropriate. Temporal replication can achieve the same reliability as spatial TMR, while reducing system hardware requirements. Algorithm-Based Fault Tolerance (ABFT), a low-overhead approach for fault detection and correction embedded in the mathematical structure of the algorithm, can be used for certain classes of applications. Other techniques, such as Software-Implemented Fault Tolerance (SIFT) and Checkpointing and Rollback, may be suitable for system-level protection.

While there will always be a tradeoff between performance and reliability, the "sweet spot" may change over the course of a mission. The goal of the proposed RFT architecture is to enable a system to adapt to the optimal balance of performance and reliability during the mission.

3.1. RFT Hardware Architecture

Two popular approaches to designing FPGA-based systems are System-on-Chip (SoC) design and coprocessor design. An SoC design uses a soft-core processor (e.g. MicroBlaze in Xilinx devices) or hard-core processors

present in the FPGA fabric (e.g. PPC405 in Xilinx Virtex-4 FX devices). These processors connect to a variety of system components using an on-chip interconnect. Application-specific modules may also be accessed by the processor using this interconnect. A coprocessor design is usually appropriate when an external processor is present or a design does not need full system-level functionality. In this methodology, control logic is implemented directly using relatively simple finite-state machines. The SoC approach is explored for the remainder of this paper, although there are no limitations preventing the use of RFT with a coprocessor approach. The SoC approach is chosen for initial evaluation because of shorter debugging and turnaround times when modifying processor code.

Figure 1 shows a high-level description of an FPGA-based SoC design. The main components are a MicroBlaze processor, a memory controller, an I/O port, and a system interconnect. Additional processing modules also connect to the system interconnect. In systems using partial reconfiguration (PR), these extra processing modules are placed in partial reconfiguration regions (PRRs).

In addition to the typical SoC components, an RFT controller is introduced to manage additional actions necessary to switch operational modes. PRRs used for processing modules connect to the RFT controller, instead of directly to the system interconnect. All components except for the PRRs can be protected using TMR, as their correct functionality is crucial to reliability of the entire system, using tools such as Xilinx's TMRTool [2] or BYU's EDIF-based TMR tool [3].

In addition to the necessary hardware modifications, the MicroBlaze processor is used to handle certain tasks required to switch states. The processor keeps track of the state of each module, including which modules are running and whether they are using TMR or a different type of mitigation scheme. Additionally, the processor communicates with the Internal Configuration Access Port (ICAP) to switch module protection schemes. When a module encounters an error, the processor can initiate recovery of the affected PRR.

3.2. Partial Reconfiguration Considerations

During the design phase of a system, a system engineer must define areas of the FPGA that will be used for partial reconfiguration. Signals that enter or leave these regions must pass through bus macros, special PR primitives used in the Xilinx PR tool flow. Additionally, the ICAP can be accessed by user logic in order to dynamically reconfigure portions of the device.

Switching between reliability modes is controlled by the processor. A signal is sent to the RFT controller to change voting procedures and to disable access to the appropriate PRRs. The necessary partial bitstreams are then sent to the ICAP to update the configuration. Finally, another signal is sent to the RFT controller to re-enable the bus connections. Switching can be triggered by external events or based upon *a priori* knowledge of the operating environment. Procedures for switching will vary on a per-mission basis.

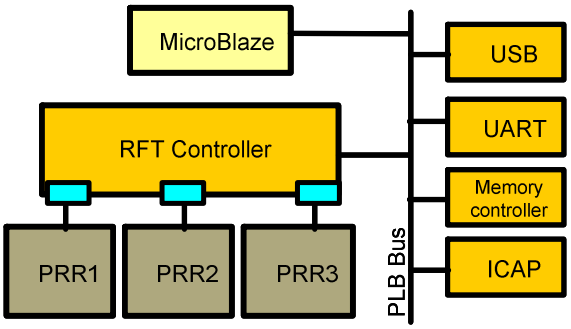

Fig. 1. Block Diagram of RFT Architecture

One benefit to the RFT approach is the need to only store partial bitstreams in a reliable memory. Each processing module has a bitstream associated for each possible PRR. Fortunately, partial bitstreams may be significantly smaller than full bitstreams, related to the size of each PRR. Storage requirements will grow linearly with the number of available PRRs, as well as with the number of possible modules. However, using a technique called bitstream relocation, it can be possible to use a single partial bitstream for a module regardless of location [10], further lowering bitstream storage requirements.

3.3. RFT Modes of Operation

There are many different FT operating modes that can be used with the RFT system. Three example modes of operation will be discussed in detail in this section. These modes represent replication-based fault mitigation, which is a subset of all the strategies that RFT encompasses.

3.3.1. Triple-Modular Redundancy Mode (RFT-TMR)

In this mode, a single processing module is replicated three times and used in three separate PRRs. The results from each module are voted on in a TMR-like fashion by the RFT controller. Any detected error signals will send an interrupt to the MicroBlaze processor, which will then initiate a recovery procedure for the faulty module. During the recovery procedure, the system state will be saved, the PRR will be reconfigured, and module state will be reloaded into the new module. This mode of operation allows for TMR of modules without requiring modifications to the original, unprotected design.

3.3.2. High-Performance Mode (RFT-HP)

In this mode, processing modules are not protected through replication. Any fault tolerance needed must be built into the module. This mode should be used when fault rates are relatively low and/or the need for performance is high. In general, data errors and many configuration errors generated while using this mode cannot be detected. When

a fault causes a module to become unresponsive, the system can force a reconfiguration of the module.

3.3.3. Self-Checking Pair Mode (RFT-SCP)

This mode of protection represents a compromise between performance and reliability. Processing modules can be replicated as self-checking pairs (SCP) instead of a full TMR approach. If a mismatch of outputs occurs, the system must reconfigure both modules and re-attempt the computation. This mode saves area in comparison to the TMR mode, allowing for the use of additional modules to increase performance. Module state can be preserved through the use of checkpointing [11].

4. MODELING APPROACH

In order to evaluate the benefit of the RFT approach to reliability, a suitable model must be created. One popular method for modeling system reliability, used frequently for reliability analysis of aerospace systems, is Markov modeling.

Traditionally, reliability analysis focuses on permanent hardware failures in systems with long lifetimes. Most available processors and FPGAs have sufficiently long lifetimes that we can ignore these failures for a short-term analysis. Instead, we are modeling computational failures induced by SEUs. For FPGA-based systems, these failures can cause either singular or multiple data errors through corrupted data or configuration memory.

Each RFT reliability mode has an associated Markov model. Each state of the Markov model corresponds to the number of operational PRRs. Transitions between states occur when a PRR changes state from operational to failed, or vice-versa. FPGA failure rates are estimated based upon available reliability data gathered by Xilinx and other sources. System repair rates are based upon the desired 'scrub' rate of the system (or system checkpointing rate).

In order to incorporate varying fault rates (due to orbit) and varying system topologies (due to RFT), a phased-mission Markov approach is used. A system is modeled as a collection of individual phases, with each phase consisting of a period of time where the RFT mode, failure rates, and repair rates are constant. The lengths of these phases are both mission-dependent and orbit-dependent. At the end of each phase, the pre-transition state probabilities are mapped onto initial probabilities for the post-transition Markov model. Figure 2 illustrates an example high-level model transitioning from TMR to SCP at time t_1, and then transitioning from SCP back to TMR at time t_2. At each phase transition, represented by the dashed vertical line, state probabilities are re-mapped.

The primary metric of interest for these models is *performability*, a combination of reliability and performance. Performability differs from availability because it provides a better metric for measuring the usefulness of a degradable system. Using the definition of performability from Eq. 1, each state in a Markov model is assigned a performance value. For this analysis, system performance is normalized to the work performed by a single PRR. Three independent PRRs running concurrently would have a performance throughput value of 3.0 while a system using three PRRs with the RFT-TMR mode would have a performance throughput value of 1.0.

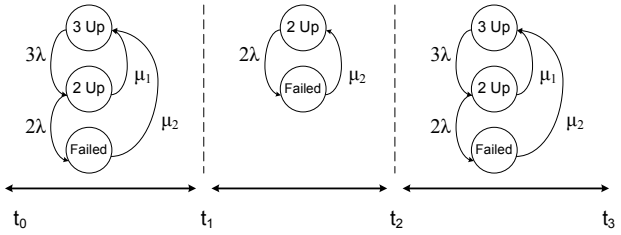

Fig. 2. Phased-Mission Markov Model

For the following sections, Markov models for each RFT mode listed in Section 3.3 are created. The solutions to the Markov models are produced using an iterative solver in MATLAB. In order to evaluate the effectiveness of each fault-tolerance strategy, instantaneous and average performability are measured.

5. RESULTS AND ANALYSIS

In this section, we present two case studies to evaluate the potential benefits of the RFT methodology. The case studies represent two possible orbits where a FPGA system might be used. The first case study represents a system operating on the International Space Station (ISS). The second case study represents an application running on a satellite in a harsher environment, a highly elliptical orbit. Each case study will compare an adaptive fault tolerance strategy to a traditional static TMR strategy.

5.1. Case Study: International Space Station

The ISS orbit is circular at an altitude of 400km and a 51.65° inclination with a mean travel time of 92 minutes. This Low Earth Orbit (LEO) avoids traveling over the Earth's poles in order to reduce the amount of radiation to which crews are subjected. Figure 3 shows the estimated number of upsets per hour that may occur in this orbit. The largest peak occurs while the ISS travels across the South Atlantic Anomaly (SAA), and the two smaller peaks occur when the ISS is closest to the poles. The upset rates used in the case studies are based upon available experimental data for Xilinx Virtex devices [12]. Determining the exact values for these rates requires significant resources, as they are both device- and orbit-dependent. Additionally, expected fault rates vary based on solar weather conditions. However, the trends presented in Figure 3 are representative of time-varying LEO upset rates, although they err on the side of being overly pessimistic. For the following figures, the data shown represents fault rates and performability over the duration of the specified orbit.

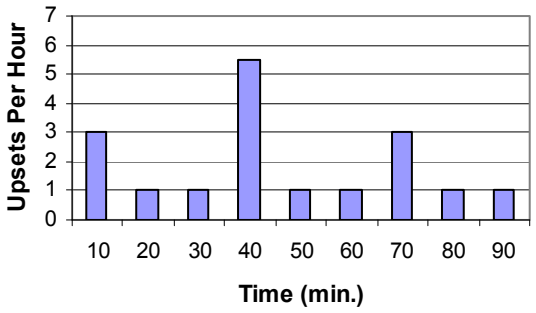

Fig. 3. ISS Orbit Fault Rates

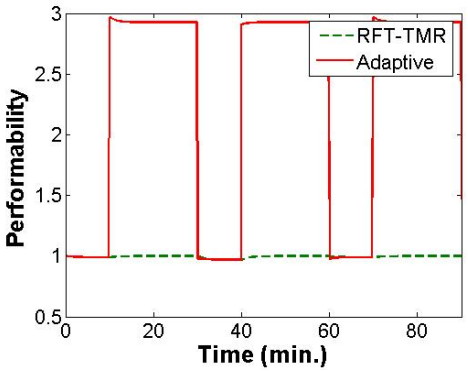

Fig. 4. ISS Orbit Performability

In order to obtain the most performance out of applications on this platform, we use the RFT-HP mode when upset rates are relatively low. During high-upset periods (SAA and poles), the system will reconfigure and use the high-reliability, RFT-TMR mode. The system will perform scrubbing to ensure that configuration memory errors are removed from the system. For the ISS model, the system uses a 30-second scrub cycle. During high-performance mode, scrubbing can correct faults that occur to the configuration memory. However, there is no mechanism for detecting that a data error has occurred.

5.2. ISS Results

Based on our model, a system using a traditional TMR approach would obtain an average performability of 0.994. Since the repair rate of the system (through scrubbing) is much larger than the expected fault rates, most configuration memory faults will be rapidly mitigated. Figure 4 shows that an RFT-based system employing RFT-TMR mode statically obtains performability during the high-radiation parts of the orbit similar to an adaptive RFT system. By contrast, during low-radiation phases, an adaptive RFT system demonstrates a significantly higher performability. The average performability of the adaptive RFT system is 2.28, a 2.3× improvement over the traditional TMR system. For applications which can tolerate a low rate of data errors without requiring redundancy (like some types of filtering or image processing applications), this operational mode can allow for large performance gains. However, other applications require that errors are, at the very least, detectable. In order to detect errors, SCP or TMR modes must be used.

5.3. Case Study: Highly Elliptical Orbit

Another common type of orbit that is used mostly by communication satellites is Highly Elliptical Orbit (HEO). From the ground, satellites traveling in an HEO orbit can appear still in the sky for long periods of time. In addition to communication satellites, HEO orbits offer visibility of the Earth's polar regions, which most geosynchronous satellites cannot. The orbit used for this case study has a perigee of 300km, an apogee of 1400km, and a 98° inclination with a mean travel time of 101 minutes. The amount of radiation observed while traversing the Earth's poles is larger than the previous case study. The average amount of radiation throughout the orbit is also increased. Figure 5 shows the estimated number of upsets per hour that an FPGA might experience in an HEO orbit [12]. The upset rates that occur during the orbit necessitate the need for redundancy at all times.

In order to obtain as much performance as possible, the adaptive system can alternate between using 3 PRRs in RFT-TMR mode and 4 PRRs running 2 applications using RFT-SCP mode. In either mode, the modules will checkpoint their state every 5 minutes. When errors are detected while using RFT-SCP mode, the FPGA will reconfigure the two affected PRRs and attempt to restart computation from a checkpoint. When using RFT-TMR mode, recovery from a checkpoint is only necessary when multiple nodes fail.

5.4. HEO Results

Figure 6 shows that the RFT-SCP and adaptive modes clearly outperform the RFT-TMR mode. However, the differences between the adaptive and RFT-SCP modes are much more subtle as they show similar average behavior. The RFT-SCP mode outperforms the adaptive mode during the high-radiation parts of the orbit due to ability to process two sets of data simultaneously. During low-radiation phases, the adaptive method outperforms RFT-SCP in performability by employing the switch from the TMR-RFT to SCP-RFT due to higher system availability at the phase transition. The adaptive RFT approach achieves a small increase in performability over solely using the RFT-SCP mode. The average performability of the adaptive system was 1.444, while the average performability of a traditional TMR system would be 0.966. A system that

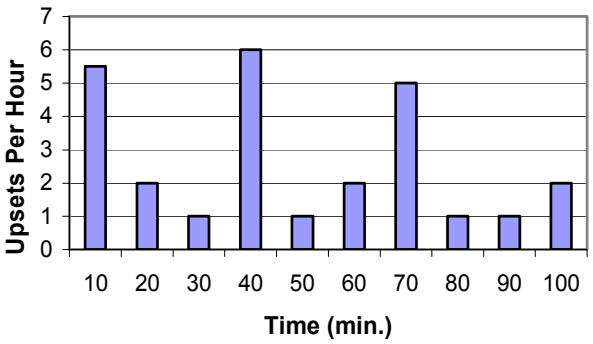

Fig. 5. Elliptical Orbit Fault Rates

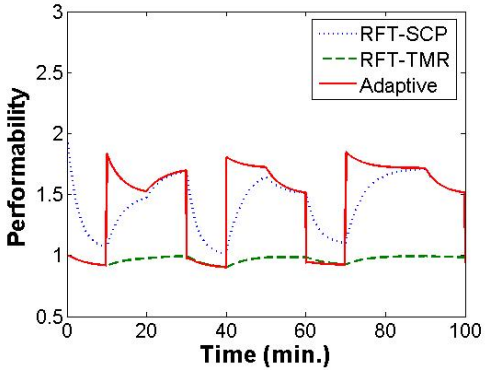

Fig. 6. Elliptical Orbit Performability

only uses the SCP mode would obtain an average performability of 1.428. Both the SCP-mode and the adaptive mode have performance benefits over the traditional TMR approach. The adaptive approach becomes more beneficial as upset rates increase in intensity and duration. The adaptive approach has an additional benefit of having an unused PRR during RFT-TMR mode. This PRR could be used to increase overall performance, or be left unused to conserve power.

6. CONCLUSIONS

A framework for enabling reconfigurable fault tolerance on Xilinx FPGA-based systems was presented and results from Markov modeling were obtained. The RFT approach provides several benefits over traditional FPGA fault tolerance. The RFT hardware architecture gives engineers the ability to provide adaptive fault tolerance without the need to rework an existing design or to only design for worst-case scenarios. By efficiently using the available resources, applications can dynamically adapt both reliability and performance depending on the operating environment. Markov models of the RFT architecture show that non-critical applications can achieve 2.3× performability improvement in relatively low-radiation environments such as the ISS orbit.

Future work includes extending the possible types of operational modes in order to support additional, low-overhead techniques such as algorithm-based fault tolerance. We also plan to explore the use of bitstream relocation to allow for hardware checkpointing and the relocation of modules to alternate PRRs during execution.

7. REFERENCES

[1] C. Carmichael, "Triple module redundancy design techniques for Virtex FPGAs," Tech. Rep., Xilinx Corporation, Nov. 2001, XAPP197.

[2] Xilinx Inc., *XTMRTool User Guide*, UG156, August 2006.

[3] B. Pratt, M. Caffrey, P. Graham, K. Morgan and M. Wirthlin, "Improving FPGA design robustness with partial TMR," *Proc. of 44th Annual IEEE Reliability Physics Symposium*, March 2006, pgs. 226-232.

[4] C. Carmichael, M. Caffrey, and A. Salazar, "Correcting single-event upsets through Virtex partial configuration," Tech. Rep., Xilinx Corporation, June 2000, XAPP216.

[5] I. Troxel, E. Grobelny, G. Cieslewski, J. Curreri, M. Fischer and A. George, "Reliable management services for COTS-based space systems and applications," *Proc. of International Conference on Embedded Systems & Applications (ESA),* Las Vegas, NV, June 26-29, 2006.

[6] K. Kim and K. Park, "Phased-mission system reliability under Markov environment," *IEEE Transactions on Reliability*, Vol. 43, No. 2, June 1994.

[7] M. Smotherman and K. Zemoudeh, "A non-homogeneous Markov model for phased-mission reliability analysis," *IEEE Transactions on Reliability*, Vol. 38, No. 5, December 1989.

[8] J. Meyer, "On evaluating the performability of degradable computing systems," *IEEE Trans. On Computers*, Vol. C-29, No. 8, August 1980.

[9] R. Smith, K. Trivedi, and A. Ramesh, "Performability analysis: measures, an algorithm, and a case study," *IEEE Transactions on Computers*, Vol. 37, Issue 4, April 1988.

[10] A. Flynn, A. Gordon-Ross and A. George, "Bitstream relocation with local clock domains for partially reconfigurable FPGAs," *Proc. of Design, Automation, and Test in Europe Conference*, Nice, France, April 20-24, 2009, to appear.

[11] D. Koch, C. Haubelt and J. Teich, "Efficient hardware checkpointing – concepts, overhead analysis, and implementation," Proc. of 2007 ACM/SIGDA 15th international Symposium on Field Programmable Gate Arrays (FPGA '07), Monterey, California,February 2007.

[12] I. Troxel and A. George, "Adaptable and autonomic management system for dependable aerospace computing," *Journal of Autonomic and Trusted Computing,*to appear.