Reliable SEU Monitoring and Recovery using a Programmable Configuration Controller

Lingkan Gong*†, Alexander Kroh*, Dimitris Agiakatsikas*, Nguyen T. H. Nguyen*,
Ediz Cetin‡ and Oliver Diessel*
* School of Computer Science and Engineering, UNSW Sydney, Australia
† School of Electrical Engineering and Telecommunications, UNSW Sydney, Australia
‡ Department of Engineering, Macquarie University, Australia

***Abstract*—FPGAs are promising candidates for computational tasks in space. However, they are susceptible to radiation-induced errors in their configuration memory. The recovery of configuration errors, either by device scrubbing or by module-based recovery, involves a series of reads and writes to the FPGA's configuration port, and is efficiently performed on-chip by a fast, flexible and reliable reconfiguration controller. In this work, we consider the reliability improvement of the recently proposed Programmable Configuration Controller (PCC), a soft reconfiguration controller that has been shown to be both fast and flexible, but whose reliability, particularly in the face of radiation-induced configuration errors, has not until now been studied. To ensure that the PCC itself is reliable, we propose the use of traditional Triple Modular Redundant (TMR) combined with a novel software-based interrupt-driven fault recovery process that leverages hardware-accelerated configuration access. We report on our design space exploration to balance the utilization, error recovery performance, and reliability of the PCC. In extremely harsh radiation environments, the Mean Time to Failure of the PCC is as high as 25 years, compared with 3.5 hours for its non-protected counterpart, and that it takes as little as 27 ms to recover from a configuration memory error affecting the PCC.**

I. Introduction

There is growing interest in using off-the-shelf SRAM-based Field Programmable Gate Arrays (FPGAs) for space missions due to their low cost, high performance, flexibility and low power. However, SRAM FPGA-based designs are susceptible to radiation-induced Single Event Upsets (SEUs) that corrupt the configuration memory [1]. Triple Modular Redundancy (TMR) is a common SEU mitigation technique that protects the user application by masking errors in one of the triplicated replicas through majority voting [1]. TMR is typically used in conjunction with either dynamic, partial reconfiguration-based modular recovery or frame-based scrubbing to correct radiation-induced configuration memory errors. Both of these approaches involve a series of reads and writes from and to an FPGA configuration port, such as the Internal Configuration Access Port (ICAP) of Xilinx FPGAs. A Reconfiguration Controller (RC) typically oversees the recovery process and is a critical component of the system. Ensuring its reliability is therefore of paramount importance in guaranteeing that the overall system operates reliably.

This research was supported in part by the Australian Research Council's Linkage (LP140100328) and Discovery (DP150103866) Projects funding schemes.

A number of investigations have studied general-purpose, fast, light-weight, and easy-to-use RCs [2], [3], [4]. For space-based FPGA systems, in particular, RC designs are also constrained by, the desire to reduce the risk of radiation-induced faults. On the one hand, the more resources used by the RC, the more area that is exposed to radiation-induced SEUs, which directly impacts the Mean Time To Failure (MTTF). On the other hand, the more configuration frames used, the longer the Mean Time To Recover (MTTR) from a configuration memory error via modular reconfiguration or scrubbing.

While a significant amount of work has focused on the reliability of the user application [1], not nearly as much effort has been directed to ensuring that the infrastructure for fault monitoring and recovery is reliable. The integrity of the entire device can be corrupted when the fault-recovery process is defeated by SEUs in the configuration memory of the RC. This work presents a reliable RC with self-recovery capabilities and low resource utilization that is programmed in C, rather than in assembly or HDL, in which bugs are more likely to be prevalent due to their greater design complexities.

To ensure that the RC meets these requirements, we propose the use of a triplicated Programmable Configuration Controller (PCC) [5]. Although implemented as a soft processor, the PCC has an Application Specific Instruction Set Processor (ASIP) architecture, which provides rapid configuration recovery of the user design, as well as of the PCC itself, while also providing software programmability and design flexibility. The design supports Xilinx FPGAs, but the idea can be readily applied to FPGAs from other vendors. While [5] focused on the performance, resource usage and flexibility of the ASIP architecture, this paper focuses on the reliability of the PCC and the fault recovery process. In particular, this paper presents:

- A novel, interrupt-driven, ASIP-based, fault recovery approach to dealing with radiation-induced configuration memory errors within the PCC;

- Design space exploration to establish the necessary PCC datapath and TMR characteristics that provide the highest reliability without sacrificing reconfiguration performance or flexibility; and
- Physical placement considerations of the TMR replicas to ensure a short fault recovery time in the PCC.

The paper is organized as follows. Section II summarizes the use cases of RCs for fault-tolerant applications and their reliability, identifying the gaps with respect to the reliability of the RC. Aiming to bridge the gap, we illustrate the ASIP architecture, TMR protection and the interrupt-based fault recovery process of the PCC in Section III. Section IV provides two case studies and analyzes the impact of PCC design parameters on its reliability. Concluding remarks are given in Section V.

II. Related Work

Various proprietary and academic RCs have been developed to meet the general needs of dynamically reconfiguring FPGAs as well as the specific requirements posed by fault-tolerant applications in harsh radiation environments. The HWICAP [6] and the SEM controller [7] are Xilinx IPs that can be used for fault recovery. HWICAP, commonly used with a soft processor such as MicroBlaze, suffers from large resource overheads and slow performance [5], while the SEM controller fails to meet flexibility needs as it does not allow alternative user-defined scrubbing functions, such as selective scrubbing [8], to be developed. Significantly, it is not known whether the MicroBlaze+HWICAP or the SEM controller have been designed to be fault tolerant. Furthermore, proprietary IPs are non-trivial to triplicate without knowing their implementations.

Several academic RCs have been proposed and are able to access the FPGA configuration resources [2], [3]. More recent work combines the built-in ECC capabilities with the hard, Processor Configuration Access Port (PCAP) of the Xilinx Zynq devices to perform fast scrubbing [4]. While previous solutions, including [5], have focused upon the performance and flexibility of the RC, beyond attempting to keep the utilization of the design small, not much attention has been given to the reliability of the controller, and by implication, the impact on the system, were it to fail.

Focusing on the reliability of the RCs themselves, in [9], [10] a self-recovering RC that has the ability to correct any errors in its configuration memory by writing a pre-stored recovery bitstream to the ICAP has been proposed. Heiner *et al.* demonstrated an internal readback scrubber by triplicating the ICAP control circuits and by implementing user memory scrubbing to recover from SEUs in the RC's BRAMs [11]. However, the RCs of the approaches detailed in [9], [10], [11] are based on the PicoBlaze processor [12], which does not have an official C compiler and must cope with a limited instruction space (1,024 words). These RCs cannot therefore readily be reprogrammed to perform new or more sophisticated fault recovery functions [5].

We propose using the PCC [5] as the RC, combined with techniques employed by other reliable soft-core processors, but by also exploiting its ASIP architecture. In particular, the interrupt-driven recovery and synchronization of the PCC is similar to that of [13]. However, the RC being recovered in this paper is the infrastructure protecting the user application, whereas the MicroBlaze being recovered in [13] is part of the user application. As such, [13] only uses an interrupt to synchronize processor state. We use the interrupt mechanism to manage the entire recovery process, and more importantly, to prioritize the recovery of the PCC and the bitstream datapath over the recovery of the user application.

III. Programmable Configuration Controller

The PCC is based on an ASIP architecture. Figure 1 depicts a block diagram of the PCC. The lightly shaded blocks are the main logic of the PCC as described in [5] (See Section III-A). In this paper, we have triplicated the PCC to protect it from SEU errors and the moderately shaded blocks illustrate the TMR voting logic (See Section III-B). The darkly shaded blocks are FPGA primitives, such as the FRAME_ECC[1] and the ICAP, that are used for fault recovery.

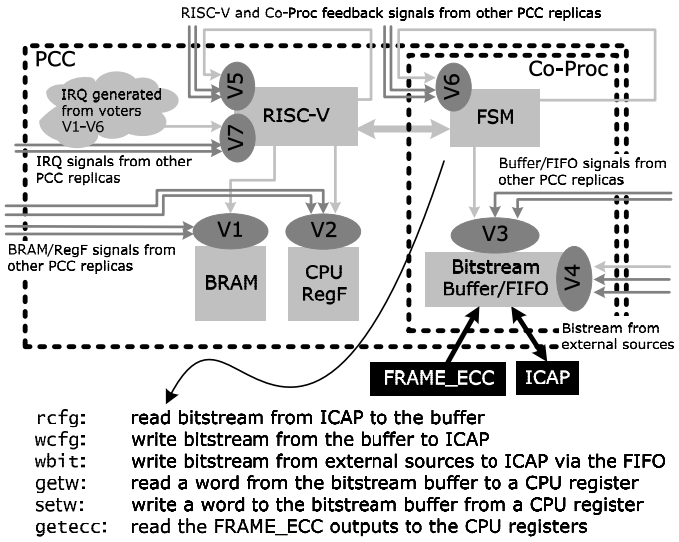


Figure 1. One copy of the triplicated PCC.

A. ASIP Architecture

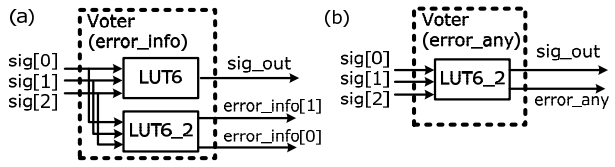
The PCC is derived from the RISC-V instruction specification and thus supports all RISC-V integer instructions and all RISC-V C compilers [14]. We removed the multiplication/division, floating point and privileged instructions since these are not required to perform reconfiguration. However, interrupt handling is an important element of the fault-recovery mechanism of the PCC (see Section III-C).

[1]FRAME_ECC is a Xilinx FPGA primitive that automatically calculates ECC data when a frame is read from the ICAP.

We have extended the RISC-V specification with the `rcfg`, `wcfg`, `wbit`, `getw`, `setw` and `getecc` instructions to gain rapid access to the configuration resources (see Figure 1 for descriptions). These instructions are implemented by a Finite State Machine (FSM) that controls a bitstream buffer, a bitstream FIFO, the ICAP port and the FRAME_ECC primitive, which are conceptually viewed as a co-processor (Co-Proc). Frame reading and writing are accomplished by a sequence of `wcfg` and `rcfg` instructions that populate the bitstream header, read/write the frame body, and write the bitstream footer. ECC-based scrubbing (e.g., [15]) can be implemented by hardware-accelerated frame accesses, as well as the `getecc` instructions. For modular recovery using partial reconfiguration (e.g., [16]), the `wbit` instruction is used to accelerate bitstream fetching. Therefore, PCC's performance is comparable to the Xilinx SEM IP but also has the advantage of being programmable to enable novel uses, such as scrubbing a region of the device in isolation of the rest [5].

B. Triplicating the PCC

We triplicated the PCC to increase its reliability by masking transient errors and mitigating the negative effects of SEUs in its configuration memory. TMR voters are inserted at a number of places in the PCC (see V1-V7 in Figure 1), and are also triplicated to avoid single points of failure. Voters 1-4 are inserted to vote on the inputs of the BRAM, Register Files and bitstream buffer/FIFO, and are referred to as boundary voters in this paper. Boundary voters isolate the memory elements and reduce the likelihood of errors being accumulated in them [17]. V7 is the boundary voter of the interrupt source of the PCC. On the other hand, synchronization voters, i.e., V5 and V6, are inserted to cut the feedback edges in the circuit topology [18]. These voters ensure that the next state, such as the next value of the Program Counter, the next state of the Co-Proc FSM, are determined by voted sources. As a result, the 3 copies of the PCC are assured of being synchronized. It should be noted that since there is no synchronization voter for the BRAM and CPU Register Files, they are explicitly synchronized after fault recovery (see Section III-C).



sig[2:0]	sig_out	error_info
000	0	11
001	0	10
010	0	01
011	1	00
100	0	00
101	1	01
110	1	10
111	1	11

sig[2:0]	sig_out	error_any
000	0	0
001	0	1
010	0	1
011	1	1
100	0	1
101	1	1
110	1	1
111	1	0

Figure 2. Two types of TMR voters.

Figure 2 illustrates two implementations of the PCC voters. The voters compare the 3 input signals and generate the boolean output that agrees with the majority of inputs, thereby masking erroneous module outputs. Traditionally, for modular recovery, a voter also generates a 2-bit "error_info" output indicating the ID of the PCC replica (i.e., 0, 1 or 2) that is in error and needs to be recovered (see Figure 2-(a)) [16]. One LUT can be used to implement an arbitrary 6-input logic function when configured in LUT6 mode, or two 5-input logic functions when configured in LUT6_2 mode [19]. Therefore, the voter in Figure 2-(a) use 2 LUTs for each bit of the signal to be voted upon. Alternatively, voters as in 2-(b) generate a 1-bit "error_any" output that indicates the existence of an error, and can thus be packed into a single LUT. Since the resource usage is halved, its reliability also improves. However, since it is not known which replica is in error, all 3 copies of the PCC have to be recovered. We analyze the impact of the voter insertion points and voter type selection in Section IV-C.

C. Interrupt-based Fault-Recovery

The PCC uses an interrupt request (IRQ) to recover from SEU errors that affect its own configuration. Figure 3 illustrates the use of the dedicated IRQ source and the execution of the Interrupt Service Routine (ISR) that reconfigures the PCC.

@t0 Let us assume that the PCC is performing an ECC-based scrub of the user application when an SEU affects replica 0 of the PCC. The error won't immediately affect the scrubbing of the user applications since the error in PCC0 is masked by the TMR voters.

@t1 Repeated errors reported by a voter cause an IRQ to be raised to the PCC. Since the interrupt sources are voted upon by V7, the voted IRQ will be asserted even if only 2 copies of the PCC are operational.

@t2 The two non-faulty replicas of the PCC accept the IRQ, suspend scrubbing the user application, and starts the ISR. The instruction fetch logic, such as the BRAM read enable, is voted upon by the V1 voter, and the fetched instructions are voted on by V5. Therefore, the correct instructions will be used in the subsequent cycles of the PCC. The ISR starts by saving the Register File contents to BRAM for future restoration. Similar to the instruction fetch, the saved register values pass through the V1 voter and thus only correct data are written to BRAM.

@t3 The PCC starts transferring the correct PCC bitstream from an external, radiation-hardened flash memory, to the ICAP. If an SEU error affects the datapath from the flash to the ICAP, the error is masked (V3 and V4) and the request to reconfigure the flash datapath is kept pending until the PCC finishes recovering itself.

@t4 After reconfiguration, the freshly reconfigured PCC replica is in an undefined state. PCC performs the state synchronization by a number of NO-OP instructions. In this period, since the pipeline registers, such as the Program Counter and the Co-Proc FSM state, etc, are voted on by V5 and V6, these registers are updated with voted, correct values. Afterwards, PCC restores the

Register Files with previously saved values, which are voted and corrected by V2. After this, all PCC replicas resume lock-step operation.

@t5 On executing a "return-from-interrupt" instruction, the global interrupt mask is automatically cleared and any pending IRQ request, such as reconfiguring the flash datapath, is accepted and served accordingly. After all the IRQs are served, the PCC resumes the ECC-based scrubbing of the user application that was suspended @t2.

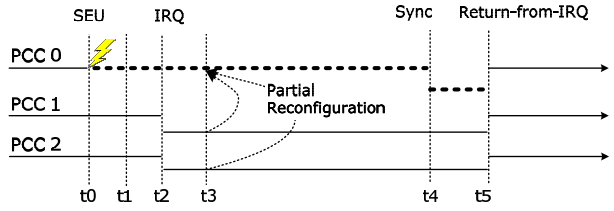


Figure 3. Interrupt-based fault-recovery.

Since the PCC has been triplicated and instructions are voted upon, the two correctly functioning copies of the PCC are able to reconfigure the faulty copy. The use of synchronization voters ensures that the state of the functioning PCC replicas is copied to the recovered replica after reconfiguration.

IV. Evaluation

We validated the triplicated PCC via a number of optimizations and two case studies. In our evaluation platform, the triplicated PCC connects to an external, bitstream storage flash via a triplicated flash controller. Since there is only one copy of the flash and only one instance of the ICAP and FRAME_ECC primitives, the voter to/from these components are not triplicated. We assume that errors in these FPGA primitives and the off-chip components require a device reset to clear them. The user application consisted of a number of synthetic TMRed compute nodes with each node containing a TMR voter that identified the ID of the erroneous module.

We present the reliability model in Section IV-A, and use the model to guide our optimization of the non-replicated, baseline PCC (Section IV-B), initial TMRed PCC (Section IV-B), and the optimized PCC (Sections IV-C and IV-D).

A. Reliability Model

We use MTTR and MTTF to evaluate the reliability of the PCC. We define an *error* as an SEU event that corrupts a configuration bit. Not all configuration bits are critical and we use *essential bit* (E-Bit) to denote bits that, when flipped, may result in observable functional impacts. We define a *failure* to be an event that prevents the PCC from functioning correctly so that it can only be recovered by a full reset. Since there is no redundancy in the non-TMR PCC, the flip of any E-Bit leads to a failure. Its MTTF is only affected by the error rate: MTTF = $1/\lambda$ [20], where λ = E-bits * *bit-error-rate*, and the *bit-error-rate* is determined by the radiation flux and the physical characteristics of the device. For the triplicated PCC, it only fails if it suffers a second error in one of the other two PCC replicas before it has recovered from the first error. Therefore, its MTTF is determined by the MTTR, apart from the error rate, λ: MTTF = $(5\lambda + (1/MTTR))/6\lambda^2$ [20].

According to the reliability model, the focus of the resource optimization is to reduce the overhead of the TMR voting logic so as to reduce the number of E-Bits. The focus of the performance optimization is to reduce the recovery (ISR) time, and thus the MTTR. This requires improving general instruction performance, as indicated by the Cycles Per Instruction (CPI) count, improving the speed of the extended reconfiguration instructions, and reducing the bitstream size of the PCC itself. We also need to ensure that the PCC operates at 100MHz so that it does not throttle the ICAP and limit reconfiguration performance. In subsequent sections, we calculate the MTTF using a worst case expected configuration memory upset rate (2.7E-10 SEUs/bit/s) for the target Xilinx Artix-7 FPGA in Geosynchronous Equatorial Orbit (GEO) [16].

B. Placement-Aware Optimizations

The placement of the PCC is constrained by the fact that on the target Artix-7 FPGA, a pblock, i.e., a rectangular placement block used to define a partially reconfigurable region, has to be a multiple of CLB columns wide, each containing ~400 LUTs. By moving non-essential components, such as the the debug UART/GPIO, outside the PCC, the PCCv2 (See Table I) uses 38% fewer LUTs as compared with the PCCv1 of [5], which results in 20% reduction in the pblock size, since the LUT reduction also falls below a 400-LUT boundary and fewer CLB columns. We subsequently used PCCv2 as the baseline design for our evaluations.

Table I also reports the results for PCCv2-TMRv1-PR, which is a triplicated PCC design, with each PCC replica mapped to a reconfigurable partition. In this design, introducing the TMR voters resulted in a 7.2X increase in resource usage and a 57% penalty to the clock frequency compared to the PCCv2 baseline. TMR voting incurred a large number of extra LUTs since there are 2,304 signals to be voted upon. Furthermore, the tool sacrificed timing so as to fully route all voting signals via the limited routing resources available across the reconfigurable partition boundaries. The results of Table I were deemed unacceptable and motivated further optimization.

C. Voter Insertion Points Analysis

Table II reports on a number of design iterations to find optimal voter insertion points. We refer to the optimized TMR voting schemes as -TMRv2 and -TMRv3, as compared to -TMRv1 of Table I.

For the -TMRv1 design, the long delay paths were those that crossed module boundaries and also happened to be on the feedback edges of the circuit. As a result, both a

Table I BASELINES OF TMR EVALUATIONS

Design	Voter location	Voter type	Resource usage (Slice/LUT/FF)	Freq (MHz)	Placement
PCCv1 [5]	non-TMR	–	573/1758/1048	100.4	5 CLB cols
PCCv2	non-TMR	–	352/1164/533	102.6	4 CLB cols
PCCv2-TMRv1-PR	all FF outputs	error_info	2488/8383/9123	43.5	10 CLB cols * 3

Table II VOTER INSERTION POINT ANALYSIS OF THE TMRED PCC

Design	Voter location	Voter type	Resource usage (Slice/LUT/FF)	Freq (MHz)	Placement	E-Bits (KBits/%)	Bitstream (KBytes)	MTTR (ms)	MTTF (years)
PCCv2	non-TMR	–	352/1164/533	102.6	4 CLB cols	299 /0.49	–	–	0.0004
PCCv2-TMRv2-PR	selected FFs	error_info	1996/5971/5484	45.5	10 CLB cols * 3	1,773/2.91	463	41.4	5.2
PCCv2-TMRv3-PR	selected FFs	error_any	1381/4994/2409	87.7	17 CLB cols	1,264/2.07	787	36.5	11.7

synchronization voter and a boundary voter were inserted on these paths, which added extra delay. In the -TMRv2 design, we moved the synchronization voters to cut different edges of the same feedback loop, or removed the boundary voters if the signals could be voted upon by downstream pipeline stages of the PCC. By carefully selecting the voter insertion points, -TMRv2 only votes on 1,227 (as opposed to 2,304) signals for all PCC replicas, and achieved an 18.4% area reduction when compared with the -TMRv1 design.

To resolve the routing congestion in the -TMRv1 design, the PCCv2-TMRv3-PR design placed all 3 PCC replicas into a single reconfigurable region so that the voting signals were internal to the pblock, where the routing requirements are relaxed. As a result, we observed a boost in the operating frequency. Furthermore, since we only have ONE reconfigurable region, it is no longer necessary for the voters to distinguish which PCC replica is in error. We could therefore use the "error_any" voters (see Figure 2-(b)), which saved area and reduced the number of E-Bits. The drawback of the -TMRv3 design is that the partial bitstream size is increased to include 3 replicas (as opposed to 1). Overall, we observed that the boost in the operating frequency and the reduction in E-Bits outweigh the negative impacts of the larger reconfiguration bitstream size, with both MTTR and MTTF improving.

D. ALU Bitwith Analysis

The largest sub-module of the PCCv2 is the 32-bit ALU. To reduce the resource usage and the number of E-Bits for the design, we re-implemented the PCC to support sub-32-bit ALU bitwidths in a way similar to [21]. Since a sub-32-bit ALU requires multiple cycles to execute a 32-bit ALU instruction, an FSM had to be introduced and the PCC was modified to become a multi-cycle processor, referred to as PCCm*X* in Table III, where the m indicates multi-cycle and *X* refers to the ALU bitwidth.

From the 32-bit-ALU to the 2-bit-ALU PCCs, we see a 21% reduction in the number of E-Bits, and a 15% reduction in the bitstream size (Table III). We also, as expected, but undesirably so, observed a steady slowdown of the PCC CPI performance, as reported by the Dhrystone benchmark. The best MTTR is achieved by a small, 8-bit ALU PCC, whereas the highest MTTF is from a 2-bit ALU PCC. This is because the ALU instructions are only used for PCC synchronization, which account for $< 0.1\%$ of total recovery time. The recovery performance primarily depends upon the speed of the reconfiguration process, which benefits from the higher clock frequency and the smaller bitstream sizes.

As illustrated in Table III, the optimized PCCs achieve an MTTF of up to 25 years and therefore could be used in GEO satellites that traditionally have a 15-year lifespan. The PCC can be safely incorporated into small Low Earth Orbit (LEO) satellites e.g. CubeSats, where the radiation level is much lower and their lifespan is usually no longer than 2 years.

E. Case Studies

We ran two user applications as case studies to validate the PCC-based recovery. Case Study 1 used module-based recovery, whereby the PCC read the configuration frame of the voter and extracted the erroneous module ID while reading. If the extracted ID indicated an error, the PCC reconfigured the erroneous module [5]. Case Study 2 used ECC-based recovery: the PCC read out the configuration frames of a module and checked the FRAME_ECC primitives while reading. If the FRAME_ECC reported an error, the corrected frame was written back to the configuration memory [5].

Table IV compares the results of the 32-bit, pipelined PCC (i.e., PCCv2-TMRv3-PR), the 8-bit, multi-cycle PCC (PCCm8-TMRv3-PR), and the baseline PCCv2. The test results of the baseline PCCv2 are the same as its predecessor, the PCCv1 in [5]. The 32-bit, pipelined PCC, PCCv2-TMRv3-PR is suited to scrubbing, since it achieves comparable performance on ECC check and correction to that of the PCCv1/PCCv2. The 8-bit ALU PCC, PCCm8-TMRv3-PR, is more suited to module-based recovery due to its higher reconfiguration throughput.

V. CONCLUSIONS AND FUTURE WORK

SRAM FPGA-based space applications are susceptible to SEU errors in the configuration memory, and the RC is the key

Table III ALU BITWIDTH ANALYSIS

Design	ALU	CPI	Resource usage (Slice/LUT/FF)	Freq (MHz)	Placement	E-Bits (KBits/%)	Bitstream (KBytes)	MTTR (ms)	MTTF (years)
PCCv2-TMRv3-PR	32	1.2	1381/4994/2409	87.7	17 CLB cols	1,264/2.07	787	36.5	11.7
PCCm16-TMRv3-PR	16	4.97	1183/4389/2643	95.6	15 CLB cols	1,109/1.81	726	30.9	17.9
PCCm8-TMRv3-PR	8	6.05	1177/4182/2622	99.6	13 CLB cols	1,036/1.70	664	27.1	23.4
PCCm4-TMRv3-PR	4	8.2	1150/4104/2613	96.9	13 CLB cols	1,041/1.70	664	27.9	22.6
PCCm2-TMRv3-PR	2	12.5	1163/4090/2610	99.3	13 CLB cols	997/1.63	664	27.2	25.2

Table IV RESULTS OF CASE STUDIES

	ALU	Freq (MHz)	Frames write/read	Recfg. Thpt. (Case Study 1)	Check a frame (Case Study 2)	Correct an error (Case Study 2)
PCCv2 (or PCCv1 [5])	32	100	3.41/3.39 us	24.4 MB/s	3.6 us	7.5 us
PCCv2-TMRv3-PR	32	87.7	3.7/3.7 us	21.6 MB/s	3.7 us	9.1 us
PCCm8-TMRv3-PR	8	99.6	5.1/5.0 us	24.5 MB/s	5.0 us	13.2 us

component that manages the detection of and recovery from SEU errors. Apart from the requirements of being fast, small and flexible, the RC must also work reliably in harsh radiation environments since a failure of the RC will leave the entire FPGA unprotected. To the best of our knowledge, the PCC proposed in this paper is the first work to use a soft ASIP architecture to implement a reliable RC that provides robust fault-recovery of SEUs in both the user logic and the RC itself.

The PCC uses well-known TMR techniques to protect its logic, and uses a novel, interrupt-based method to synchronize the freshly recovered copy of the PCC with the other two operating copies. Through a careful analysis, we have optimized the placement, voter insertion points and the ALU bitwidths of the PCC. Our evaluations indicate that placing 3 PCC replicas in a single reconfigurable partition is more reliable than placing them each in their own partition. We have also identified that the fastest MTTR is achieved by using an 8-bit ALU, which recovers itself in 27ms, whereas the highest MTTF is achieved by the 2-bit ALU PCC, which can run 25.2 years without failure in GEO orbit, achieving 63,000X improvement in MTTF over the non-TMR PCC. PCC is thus a tiny, software-programmable, high performance and, most importantly, highly reliable reconfiguration controller.

Looking ahead, we intend to further study the relationship between PCC reliability and system-wide reliability. We also plan to conduct fault-injection experiments in order to validate the sensitivity to faults of the triplicated PCC.

REFERENCES

[1] F. Siegle, T. Vladimirova, J. Ilstad, and O. Emam, "Mitigation of Radiation Effects in SRAM-Based FPGAs for Space Applications," *ACM Computing Surveys*, vol. 47, no. 2, pp. 37:1–37:34, 2015.

[2] K. Vipin and S. A. Fahmy, "A High Speed Open Source Controller for FPGA Partial Reconfiguration," in *Int. Conf. on Field-Programmable Technology (FPT)*, 2012, pp. 61 – 66.

[3] S. D. Carlo, P. Prinetto, and P. Trotta, "A Portable Open-Source Controller for Safe Dynamic Partial Reconfiguration on Xilinx FPGAs," in *Int. Conf. on Field Programmable Logic and Applications (FPL)*, 2015, pp. 1 – 4.

[4] A. Stoddard, A. Gruwell, P. Zabriskie, and M. Wirthlin, "High-speed PCAP Configuration Scrubbing on Zynq-7000 All Programmable SoCs," in *Int. Conf. on Field Programmable Logic and Applications (FPL)*, 2016, pp. 1–8.

[5] L. Gong, T. Wu, N. T. H. Nguyen, D. Agiakatsikas, Z. Zhao, E. Cetin, and O. Diessel, "A Programmable Configuration Controller for Fault-Tolerant Applications," in *Int. Conf. on Field-Programmable Technology (FPT)*, 2016, pp. 117–124.

[6] Xilinx, *AXI HWICAP (PG134)*, 2015.

[7] ——, *Soft Error Mitigation Controller (PG036)*, 2015.

[8] D. Agiakatsikas, E. Cetin, and O. Diessel, "FMER: A Hybrid Configuration Memory Error Recovery Scheme for Highly Teliable FPGA SoCs," in *Int. Conf. on Field Programmable Logic and Applications (FPL)*, 2016, pp. 1–4.

[9] A. Ebrahim, K. Benkrid, X. Itrube, and C. Hong, "A Novel High-Performance Fault Tolerant ICAP Controller," in *NASA/ESA Conf. on Adaptive Hardware and Systems*, 2012, pp. 259–263.

[10] A. Ebrahim, T. Arslan, and X. Iturbe, "On Enhancing the Reliability of Internal Configuration Controllers in FPGAs," in *NASA/ESA Conf. on Adaptive Hardware and Systems (AHS)*, 2014, pp. 83–88.

[11] J. Heinerl, N. Collins, and M. Wirthlin, "Fault Tolerant ICAP Controller for High-Reliable Internal Scrubbing," in *IEEE Aerospace Conf.*, 2008, pp. 249–258.

[12] Xilinx, *PicoBlaze 8-bit Microcontroller User Guide (UG129)*, 2011.

[13] Y. Ichinomiya, S. Tanoue, M. Amagasaki, M. Iida, M. Kuga, and T. Sueyoshi, "Improving the Robustness of a Softcore Processor against SEUs by Using TMR and Partial Reconfiguration," in *IEEE Symp. on Field-Programmable Custom Computing Machines (FCCM)*, 2010, pp. 47 – 54.

[14] A. Waterman, Y. Lee, D. Patterson, and K. Asanovic, *The RISC-V Instruction Set Manual Volume I: User-Level ISA*, 2014. [Online]. Available: https://riscv.org/specifications/

[15] I. Herrera-Alzu and M. Lopez-Vallejo, "Design Techniques for Xilinx Virtex FPGA Configuration Memory Scrubbers," *IEEE Trans. on Nuclear Science*, vol. 60, no. 1, pp. 376–385, 2013.

[16] D. Agiakatsikas, N. T. H. Nguyen, Z. Zhao, T. Wu, E. Cetin, O. Diessel, and L. Gong, "Reconfiguration Control Networks for TMR Systems with Module-based Recovery," in *IEEE Symp. on Field-Programmable Custom Computing Machines (FCCM)*, 2016, pp. 88–91.

[17] B. Pratt, M. Caffrey, D. Gibelyou, P. Graham, K. Morgan, and M. Wirthlin, "TMR with More Frequent Voting for Improved FPGA Reliability," in *Int. Conf. on Engineering of Reconfigurable Systems and Algorithms (ERSA)*, 2008, pp. 1–8.

[18] J. M. Johnson and M. J. Wirthlin, "Voter Insertion Algorithms for FPGA Designs Using Triple Modular Redundancy," in *ACM/SIGDA Int. Symp. on Field Programmable Gate Arrays (FPGA)*, 2010, pp. 1–10.

[19] *7 Series FPGAs Configuration Logic Block (UG474)*, Xilinx Inc., 2013.

[20] Z.-M. Wang, L.-L. Ding, Z.-B. Yao, H.-X. Guo, H. Zhou, and M. Lv, "The Reliability and Availability Analysis of SEU Mitigation Techniques in SRAM-based FPGAs," in *European Conf. on Radiation and Its Effects on Components and Systems*, 2009, pp. 497–503.

[21] Y. Tanaka, S. Sato, and K. Kise, "The Ultrasmall Soft Processor," *ACM Computer Architecture News*, vol. 41, no. 5, pp. 95–100, 2013.