Reliability of a Softcore Processor in a Commercial SRAM-Based FPGA*

Nathaniel H. Rollins
NSF Center for High-Performance Reconfigurable Computing (CHREC)
Brigham Young University
Provo, UT.
nhrollins@byu.edu

Michael J. Wirthlin
NSF Center for High-Performance Reconfigurable Computing (CHREC)
Brigham Young University
Provo, UT.
wirthlin@ee.byu.edu

ABSTRACT

Softcore processors are an attractive alternative to using radiation-hardened processors in space-based applications. Unlike traditional processors however, the logic and routing of a softcore processor are vulnerable to the effects of single-event upsets (SEUs). This paper applies two common SEU mitigation techniques, TMR with checkpointing and DWC with checkpointing, to the LEON3 softcore processor. The improvement in reliabilty over an unmitigated version of the processor is measured using three metrics: the architectural vulnerability factor (AVF), mean time to failure (MTTF), and mean useful instructions to failure (MuITF). Using configuration memory fault injection, we found that DWC with checkpointing improves the MTTF and MuITF by over 35×, and that TMR with triplicated input and outputs improves the MTTF and MITF by over 6000×.

Categories and Subject Descriptors

B.8.1 [**Performance and Reliability**]: Reliability, Performance, and Fault-Tolerance

Keywords

Reliability, Softcore processors, AVF, MTTF, MuITF

1. INTRODUCTION

Microprocessors used in space-based applications must be protected from the effects of high-energy particles. They are usually protected through a very expensive process called radiation-hardening. Radiation-hardened processors are built with special design libraries and fabrication processes that are more resilient to high-energy radiation [1]. These larger transistors resist high-energy particles by requiring more energy to switch. Although tolerant to radiation, rad-hard processors are larger, slower, consume more power, and can cost hundreds of thousands of dollars [1,2] when compared to a commercial processor. Additionally, rad-hard processors used in space are often one to two decades old [2].

As an alternative to an older expensive rad-hard processor, processors can be implemented in the logic of a field-programmable gate array (FPGA). A processor implemented in an FPGA is called a *softcore* processor because the logic is reconfigurable. In contrast to rad-hard processors, softcore processors are fast, flexible, less expensive, and reconfigurable. If a softcore processor can be adequately protected from soft-errors, softcore processors can be an attractive alternative for space-based applications.

The flexibility and reprogramability that FPGAs provide for space-based applications comes at a price – SRAM-based FPGAs are inherently sensitive to the effects of faults caused by high-energy particles. These single event upsets (SEUs) can occur not only in FPGA user memory bits but also in the FPGA configuration bits that define the logic and routing of the softcore processor. SEUs in the FPGA configuration memory remain until repaired with configuration scrubbing.

Although FPGA user memory is also sensitive to the effects of SEUs, there are far more configuration bits in an FPGA device than user memory bits. For the device used in this study (Xilinx Virtex4 FX60), there are almost 21 million configuration bits, which is over 4× more bits than user memory bits. Configuration bits control slices (including flip-flops, LUT-RAMs, and SRLs), IOBs, DCMs, DSPs, BRAM interconnect, all instance attributes, and all routing. User memory bits include BRAM data bits and user flip-flops. Thus SEUs are more likely to occur in the logic and routing (configuration memory) than in user memory.

This paper characterizes the sensitivity of a mitigated softcore processor in the presence of configuration SEUs using a novel fault injection technique. The mitigation techniques include triple modular redundancy (TMR), and duplication with compare (DWC) with checkpointing. TMR is used since it is one of the most popular FPGA design mitigation techniques [3], and DWC with checkpointing is used since it is one of the most popular processor mitigation techniques [4]. The metrics used to characterize the reliability of the softcore processor designs include architectural vulnerability factor (AVF) [5], mean-time to failure (MTTF), and mean useful instructions to failure (MuITF) [6].

*This work was supported in part by the I/UCRC Program of the National Science Foundation under the NSF Center for High-Performance Reconfigurable Computing (CHREC) under Grant No.0801876.

FPGA'12, February 22–24, 2012, Monterey, California, USA.

2. SOFTCORE PROCESSOR DESIGNS

The processor used in this study is Aeroflex Gaisler's 32-bit LEON3 processor [7]. This processor is chosen for this study because of its popularity in the space community, and because it is open-source. Only the core microarchitectural components of the LEON3 processor are included in this study. Figure 1 shows that the core units include the 7-stage integer pipeline, a 12 window register file, the hardware multiplier and divider, 1 Kbyte direct-mapped instruction and data caches, interrupt controller, and on-chip main memory.

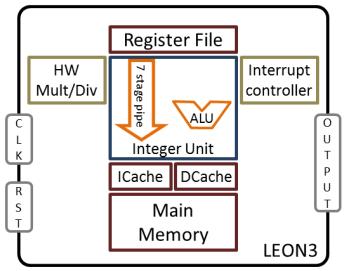

Figure 1: The LEON3 core processor units used in this study.

The first mitigation technique used in this study to protect the LEON3 processor uses duplication with compare (DWC) and rollback checkpointing [8]. DWC is a reliability technique that uses a duplicate processor to detect upsets. This study includes two versions of the DWC LEON3 processor: one with duplicated clock and reset inputs, and one with single clock and reset signals. Duplication is a popular processor reliability technique that can either be used in lockstep [8] or dual-core execution (DCE) [9]. This study uses duplicated processors in lockstep.

Softcore processors have the luxury of using strict lockstep since *any* processor signal can be exposed on a cycle-by-cycle basis. The softcore processor architecture can be changed at any time to use any given signal for lockstep comparison. The lockstep DWC processors compare register file outputs, program counters, instruction registers, processor output, and main memory input in a strict lockstep fashion.

The second LEON3 processor mitigation technique uses TMR and roll-forward checkpointing [8]. This study uses two versions of the TMR LEON3 processor: one with triplicated inputs and outputs (with off-chip voting), and one with untriplicated inputs and outputs (with on-chip voting). Similar to the duplicated processor design, the TMR design keeps the triplicated processors in strict lockstep execution using the register file outputs, program counters, instruction registers, processor outputs, and main memory inputs. But unlike DWC design, when one of the processors falls out of lockstep, a rollback is not required. Instead, the state of the two processors still in lockstep is used to correct and resynchronize the one that is out of lockstep.

3. RELIABILITY METRICS

Traditional processor reliability is measured in terms of mean time to failure (MTTF) or more recently, mean instructions to failure (MITF) [6]. This section shows how these metrics are modified for softcore processors. The metrics introduced in this section can be applied to the processor as a whole or to individual components of the processor to compare component reliability.

Not every configuration upset in a softcore processor will lead to erroneous processor output. The traditional metric for expressing how likely it is for an upset to lead to a processor error is called the *architectural vulnerability factor* (**AVF**) [5]. When applied to softcore processors, AVF is used to represent the percentage of configuration memory bits that, when upset, cause the processor to execute incorrectly.

Each of the 21 million bits in the configuration memory is classified as being either required for architecturally correct execution (**ACE**) or unnecessary for architecturally correct execution (**unACE**) [5]. Upsetting an ACE bit causes a program running on the softcore processor to produce an incorrect output, while upsetting an unACE bit does not hinder correct program execution. In this study the terms ACE bits and *sensitive bits* are used synonymously. Likewise, unACE is a synonym for *insensitive*.

In keeping with the meaning of a sensitive bit, upsets to ACE bits are classified as being either detected, recoverable upsets (**DRU**), detected, unrecoverable errors (**DUE**), or silent data corruption (**SDC**) bits. DUE upsets are those which are detected by the processor, but from which the processor cannot recover. SDC upsets are upsets which are never detected, and which cause erroneous output. In order to have a reliable processor, there should be as few SDCs and DUEs as possible. SDCs are especially bad since they are not even detected. When mitigation techniques are used, the number of DRU bits indicate how many upsets are detected and prevented from causing erroneous output. Mitigation techniques attempt to reduce the number DUEs and SDCs and increase the number of DRUs.

Processor SEU sensitivity is measured using AVF. The AVF of a processor is the percentage of the processor area that is sensitive to DUEs and SDCs [5]. Equation 1 shows how AVF is computed for softcore processors. AVF is the fraction of the configuration bits (CFGbits) that contain SDC or DUE bits.

$$\text{AVF} = \frac{\#\text{ SDCs} + \#\text{ DUEs}}{\text{CFGbits}} \tag{1}$$

AVF can be used to estimate the reliability in terms of mean time to failure (MTTF). MTTF is computing using the configuration upset rate (λ) with AVF: $\text{MTTF} = \frac{1}{\lambda} \cdot \frac{1}{\text{AVF}}$. The upset rate of the processor is equal to λ_{bit} multiplied by the number of configuration bits used by the processor ($\lambda_{proc} = \lambda_{bit} \cdot$ CFGbits). λ_{bit} depends upon the spacecraft orbit, space environment, and device properties. For this paper, λ_{bit} is estimated using a galactic cosmic ray (GCR) environment at solar minimum. The estimated upset rate is 1E-10 upsets/bit-day [10] or 1.16E-15 upsets/bit-s. The MTTF for the processor or a component is:

$$\begin{aligned} \text{MTTF} &= \frac{1}{\lambda_{bit}} \cdot \frac{1}{\text{CFGbits}} \cdot \frac{1}{\text{AVF}} \\ &= \frac{1}{\lambda_{bit}} \cdot \frac{1}{\#\text{ SDCs} + \#\text{ DUEs}}. \end{aligned} \tag{2}$$

Although MTTF provides a reasonable reliability measure, it does not account for the processor performance costs. Instead of measuring the time between two errors, mean instructions to failure (MITF) measures the amount of work

accomplished between two errors [6]. MITF expresses how many instructions a processor commits, on average, between two errors. When mitigation techniques are used, MITF must also account for any performance costs (ρ) incurred from the mitigation techniques. The performance cost represents the execution of additional *unuseful* instructions. Equation 3 shows how the mean *useful* instructions to failure (MuITF) is computed for a processor:

$$\begin{aligned} \text{MuITF} &= \frac{\text{frequency}}{\lambda_{bit}} \cdot \frac{1}{\text{CFGbits}_{mit}} \cdot \frac{\text{IPC}}{\text{AVF} \cdot \rho} \\ &= \frac{\text{frequency}}{\lambda_{bit}} \cdot \frac{\text{IPC}/\rho}{\#\text{ SDCs} + \#\text{ DUEs}}. \end{aligned} \quad (3)$$

4. HARDWARE FAULT-INJECTION

To evaluate the reliability of the mitigated and unprotected LEON3 processor designs, hardware fault-injection is used to identify ACE bits in the configuration memory. Traditionally, ACE (and unACE bits) bits only occur in user memories and registers. ACE bits are normally identified by tracking them in the pipeline or with the use of models [11]. This section describes the hardware fault-injection procedure that identifies ACE bits in the configuration memory.

Hardware fault-injection is a well-known way of evaluating the impact of SEUs on commercial SRAM-based FPGA devices [12]. Fault-injection is performed by upsetting each and every bit in the FPGA configuration memory, one at a time, while a program executes on the LEON3 processor. For the Xilinx Virtex4 FX60 FPGA used in this study, almost 21 million upsets and program executions are required to test every bit in the entire configuration memory.

For each of the 21 million configuration bits in a mitigated LEON3 processor, a novel fault-injection procedure is followed. Overall, the program running in the LEON3 processor on the DUT FPGA runs four times for each of the 21 million configuration memory bits. The program runs once to ensure that at least one checkpoint is taken. In the next program iteration, the program runs for a random number of clock cycles before the configuration bit is upset. After the upset is inserted, the second program execution finishes and then runs a third time. This gives the LEON3 ample time to detect the upset. If the upset is detected, the upset is immediately repaired by the fault-injector. If the upset is detected, the DUT also attempts to recover from the upset with a checkpoint rollback. If the upset goes undetected by the end of the third program execution, it is repaired. Finally, the program runs for a fourth time to ensure that any checkpoint recovery attempted by the DUT is successful. A similar process is followed to test the configuration bits in the unmitigated LEON3, but only the middle two program runs are required.

The full fault-injection procedure is run for all 21 million configuration bits for eight micro-benchmarks used in this study. The benchmarks used in this work are limited by the constraints of the fault injector and by the number of BRAMs available on the FPGA. Running the fault-injection procedure on a LEON3 design with one of these micro-benchmark programs takes up to 60 hours to complete. The benchmark programs in this study test worst-case behavior of the processor structures and instruction set architecture. Since these benchmarks are meant to be stress tests, it is expected that they will be pessimistic compared to a typical workload.

5. PROCESSOR COMPARISONS

This section discusses the reliability-cost trade-off for applying mitigation to the LEON3 softcore processor. The costs are measured in terms of area and performance with respect to an unmitigated processor. Reliability is measured using AVF, MTTF, and MuITF.

The area costs of the unmitigated LEON3 processor are compared with the mitigated LEON3 designs in Table 1. Area is compared in terms of slices, BRAMs, DSPs, and configuration memory bits (CFGbits). The values in braces show the area increase (in terms of configuration bits) compared to the unmitigated processor. It shows that the area cost of DWC is significantly more than double and that the area cost of TMR is more than triple.

Area Costs				
LEON3 Design	Slices	BRAM	DSP	Config Bits (CFGbits)
Unmitigated	3058	12	4	677,065 (1.00×)
DWC (1 clock)	8628	33	8	1,834,002 (2.71×)
DWC (2 clocks)	8569	33	8	1,837,719 (2.71×)
TMR (1 in/out)	14,623	36	12	3,102,921 (4.58×)
TMR (3 in/out)	14,286	36	12	3,075,399 (4.54×)

Table 1: Area cost comparison for the LEON3 processor designs.

The addition of checkpointing to the LEON3 processor introduces a small performance cost (ρ). The value of this cost is proportional to the frequency with which checkpointing occurs. In this study, checkpointing occurs once per program execution. The performance cost incurred is reported in terms of the additional number of clock cycles needed, on average, to run a program compared to when running on an unmitigated processor. On average, programs on the DWC processors run 1.01× longer than when run on an unmitigated processor. The time to record checkpoint information accounts for the small performance penalty incurred by the DWC designs. The performance cost for the TMR processor is negligible since the only incurred performance cost comes when roll-forward checkpointing corrects and resynchronizes one of the three processors.

Hardware fault-injection is used to measure the total number of ACE and unACE bits for each of the LEON3 softcore processor designs and with each of the benchmark programs (Table 2). The percentage in the unACE column indicates the percentage of used configuration bits that, when upset, do not hinder correct program execution. The percentages reported in the DRE, DUE, and SDC columns are with respect to only the ACE bits for the given processor.

Table 2 shows that both mitigation techniques significantly reduce the number of SDCs and DUEs of the unmitigated processor, but that TMR with triplicated inputs and outputs almost eliminates SDCs and DUEs. In the TMR design with untriplicated inputs and outputs, 66% of the SDCs and DUEs occur in the untriplicated output, 21% occur in the clock and reset signals, and 12% occur in the voters. In the DWC designs, a large majority of the SDCs and DUEs occur in the top-level outputs, clock and reset signals, or in the comparator unit.

Table 2 also shows that some of upsets are detected by the unmitigated LEON3 processor (as shown in the DUE

HW Fault-Injection Results				
LEON3 Design	unACE	ACE		
		DRU	DUE	SDC
Unmitigated	537,825 (79.4%)	0 (0.0%)	11,637 (8.36%)	127,603 (91.64%)
DWC & Check (1 clock)	1,378,813 (75.2%)	450,527 (98.98%)	762 (0.16%)	3900 (0.86%)
DWC & Check (2 clocks)	1,369,864 (74.5%)	463,957 (99.17%)	376 (0.08%)	3522 (0.75%)
TMR (no triplicated in/out)	2,458,337 (79.2%)	642,523 (99.68%)	73 (0.01%)	2061 (0.31%)
TMR (triplicated in/out)	2,421,376 (78.7%)	654,013 (99.996%)	13 (0.002%)	10 (0.002%)

Table 2: Full fault-injection results for the LEON3 processors.

column). Although no upset detection techniques have been explicitly applied, the LEON3 processor pipeline has some built-in error detection. The processor throws an error signal when interrupts occur in an unexpected way.

The measured number of ACE bits are used to calculate the AVF, MTTF, and MuITF of each LEON3 processor. The reliability of each of the LEON3 processors is shown in Table 3. The average number of instructions per clock cycle (IPC) – required by MuITF (Equation 3) is 0.62. The MuITF frequency value used in Equation 3 is 33 MHz, since that is the frequency at which our fault-injection hadware runs. The AVF values in the table show that the unmitigated LEON3 is almost 100× more vulnerable than the LEON3 protected with DWC and checkpointing, and over 27,000× more vulnerable than the LEON3 protected with full TMR and roll-forward checkpointing. The reliability results show that although full TMR provides the best protection, DWC and checkpointing may be an acceptable lower-cost alternative.

LEON3 Processor Reliability			
LEON3 Design	AVF	MTTF (years)	MuITF $\times 10^{20}$ (instructions)
Unmitigated	20.6%	0.83 (1.00×)	5.36 (1.00×)
DWC (1 clock)	0.25%	24.8 (29.9×)	158.6 (29.6×)
DWC (2 clocks)	0.22%	29.7 (35.7×)	189.7 (35.4×)
TMR (1 in/out)	0.07%	55.2 (66.4×)	356.0 (66.4×)
TMR (3 in/out)	0.00075%	5032 (6058×)	32,469 (6058×)

Table 3: Comparison of AVF, MTTF, and MuITF of an unmitigated LEON3 against the mitigated LEON3 processors.

6. CONCLUSION

This study demonstrates the improvements in reliability by applying DWC and TMR with checkpointing to a softcore processor. Three metrics were used to compare an unmitigated softcore processor with these two mitigated designs: architectural vulnerability factor (AVF), mean time to failure (MTTF), and mean useful instructions to failure (MuITF). The AVF, MTTF, and MuITF were measured through hardware fault-injection. The reliability of this unmitigated processor is improved by applying DWC with checkpointing and TMR with checkpointing to the processor. DWC with checkpointing is shown to improve the MTTF and MuITF by over 35×. TMR with checkpointing improves the MTTF and MuITF by over 6000×.

This study shows that although softcore processors are sensitive to the effects of SEUs, the faults can be characterized and protected with appropriate mitigation techniques. An adequately protected softcore processor is an attractive alternative to a rad-hard processor for space-based applications. Compared to rad-hard processors, softcore processors are faster, flexible, less expensive, and reconfigurable.

7. REFERENCES

[1] Q. Zhou, K. Mohanram, Gate sizing to radiation harden combinational logic, Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on 25 (1) (2006) 155 – 166.

[2] J. F. Bell, et al., Mars reconnaissance orbiter mars color imager (MARCI): Instrument description, calibration, and performance, Journal of Geophysical Research 114.

[3] L. Sterpone, M. S. Reorda, M. Violante, F. L. Kastensmidt, L. Carro, Evaluating different solutions to design fault tolerant systems with SRAM-based FPGAs, Journal of Electronic Testing: Theory and Applications 23 (2007) 47–54.

[4] A. Ziv, J. Bruck, Analysis of checkpointing schemes with task duplication, Computers, IEEE Transactions on 47 (2) (1998) 222 –227.

[5] S. S. Mukherjee, et al., A systematic methodology to compute the architectural vulnerability factors for a high-performance microprocessor, Microarchitecture, IEEE/ACM International Symposium on 0 (2003) 29.

[6] C. Weaver, et al., Techniques to reduce the soft error rate of a high-performance microprocessor, SIGARCH Comput. Archit. News 32 (2004) 264–.

[7] J. Gaisler, E. Catovic, Multi-Core Processor Based on LEON3-FT IP Core (LEON3-FT-MP), in: DASIA 2006 - Data Systems in Aerospace, Vol. 630 of ESA Special Publication, 2006.

[8] D. Pradhan, N. Vaidya, Roll-forward and rollback recovery: performance-reliability trade-off, in: Fault-Tolerant Computing, 1994. FTCS-24. Digest of Papers., Twenty-Fourth International Symposium on, 1994, pp. 186 –195.

[9] H. Zhou, A case for fault tolerance and performance enhancement using chip multi-processors, Computer Architecture Letters 5 (1) (2006) 22 –25.

[10] R. Hillman, et al., Space processor radiation mitigation and validation techniques for an 1,800 MIPS processor board, in: Radiation and Its Effects on Components and Systems, 2003. RADECS 2003. Proceedings of the 7th European Conference on, 2003, pp. 347 – 352.

[11] S. S. Mukherjee, M. Kontz, S. K. Reinhardt, Detailed design and evaluation of redundant multithreading alternatives, Computer Architecture, International Symposium on 0 (2002) 0099.

[12] E. Johnson, M. Caffrey, P. Graham, N. Rollins, M. Wirthlin, Accelerator validation of an FPGA SEU simulator, Nuclear Science, IEEE Transactions on 50 (6) (2003) 2147–2157.