Chapter 5
A Fault Injection System for Measuring Soft Processor Design Sensitivity on Virtex-5 FPGAs

Nathan A. Harward, Michael R. Gardiner, Luke W. Hsiao, and Michael J. Wirthlin

Abstract This paper presents an FPGA fault injection system, a methodology for soft processor fault injection, and fault injection experimental results for MicroBlaze and LEON3 soft processor designs. The Xilinx Radiation Test Consortium—Virtex 5 Fault Injector (XRTC-V5FI) was built to evaluate the configuration memory sensitivity of soft processor designs. To overcome some of the challenges of soft processor fault injection, we designed the XRTC-V5FI to be fast, flexible, and to fully cover all configuration memory bits. The minimum time to inject a full bitstream is 28 minutes and the individual fault injection can be as fast as 49 μS. The LEON3 has 81.3 % more sensitive bits than the MicroBlaze, yet when normalized by the number of used slices, the MicroBlaze is 26.2 % more sensitive than the LEON3.

5.1 Introduction

Operating microelectronic devices in high radiation environments greatly increases their potential to malfunction. Energized ions colliding with sensitive logic regions within a microelectronic device can change the state of the circuit [1]. When a collision event modifies the state of a memory bit or flip-flop, this is known as a soft error or a single event upset (SEU).

Protection against SEUs is commonly achieved through the use of radiation-hardened components. However, these components are expensive and lag several generations behind standard commercial components due to high development and

N.A. Harward (✉) • M.R. Gardiner • L.W. Hsiao • M.J. Wirthlin
Department of Electrical and Computer Engineering, NSF Center for High Performance Reconfigurable Computing (CHREC), Brigham Young University, Provo, UT, USA
e-mail: nateharward@ieee.org; mikegardiner@byu.edu; lukehsiao@byu.edu; wirthlin@byu.edu

F. Kastensmidt, P. Rech (eds.), *FPGAs and Parallel Architectures for Aerospace Applications*, DOI 10.1007/978-3-319-14352-1_5

testing costs and limited production volume [1]. Field Programmable Gate Arrays (FPGAs) provide a computing platform which is a suitable and flexible alternative to radiation-hardened computers. FPGA reconfigurability allows design upgrades and corrections after a space launch, and the same FPGA can be reused for new designs.

SRAM-based FPGAs use static random-access memory (SRAM) to hold the FPGA configuration and their SRAM is vulnerable to SEUs. A change to a configuration memory bit can affect the function of a look-up table (LUT) or the routing between nodes, and cause failure in the user design. One example of such failure is illustrated in Figs. 5.1, 5.2, 5.3 and 5.4. Figure 5.1 shows the configuration memory that defines a simple circuit within an FPGA and Figure 5.2 shows the routing and logic result of that memory as an AND gate with two inputs. Figure 5.3 shows an SEU routing one of the inputs away from the AND gate and Figure 5.4 shows an SEU changing the AND gate into an XOR gate. The configuration memory on an FPGA can be protected from SEUs with memory scrubbing and/or error detection and correction (EDAC) techniques [2]. FPGA fault-tolerant design techniques such as triple modular redundancy (TMR) can also be employed to detect and mitigate SEUs.

FPGA fault injection is an emulation-based method for discovering which of the configuration bits in a design are sensitive to upset. It can help identify specific system failure modes and determine design vulnerabilities. To determine which configuration bits are sensitive, each bit is changed one by one to emulate an SEU while the design outputs are compared with outputs from a golden model or set of expected outputs. Each changed bit is restored when the next bit is changed to emulate an SEU. When an output mismatch is observed, the fault injector logs the changed bit as a sensitive bit. FPGA fault injection does not completely evaluate the reliability of a design, as it does not test all FPGA components and hard logic. FPGA fault injection does not emulate single event transients (SETs) or multi-bit errors (MBUs).

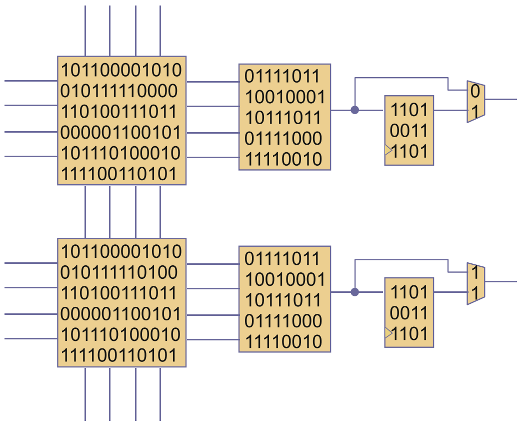

Fig. 5.1 Configuration memory

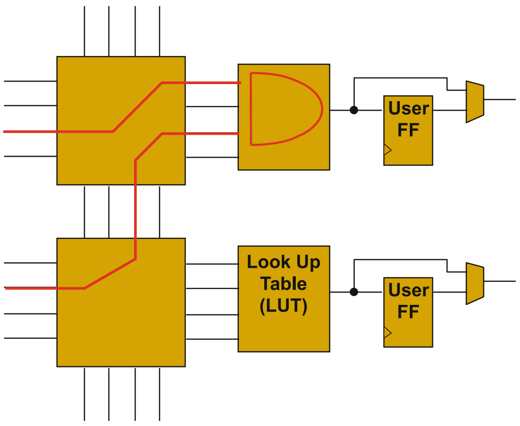

Fig. 5.2 Routing and logic result of configuration memory

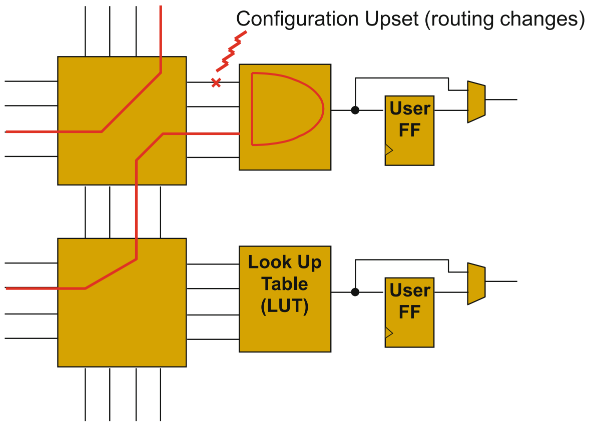

Fig. 5.3 Upset in routing

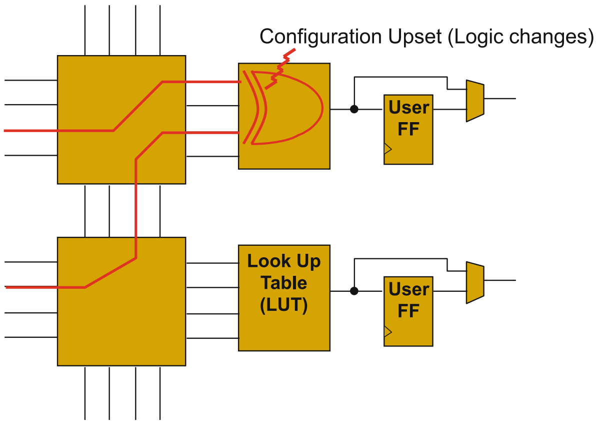

Fig. 5.4 Upset in logic

5.2 Related Works

The need for reliable FPGAs in space environments has motivated the development of FPGA fault injection platforms [2]. Over the years, many notable fault injection tools and platforms were created [3–5]. Johnson et al. used a SLAAC-1V testbed which housed three Virtex (XCV1000) FPGAs [6]. The SLAAC-1V injector was able to test all configuration bits at high speeds and predict where upsets can occur. Alderighi et al. [7] created the FLIPPER fault-injection platform which used a single Virtex-II Pro (XC2VP20) motherboard test fixture that could also be used for radiation tests. Rather than test all configuration bits, they used a probabilistic model to determine design sensitivity. Sterpone et al. [8] used a Virtex-II Pro (XCV2P30) FPGA with an embedded PowerPC microprocessor. Using an internal configuration access port (ICAP), a timing unit, and having the test design internal to the test FPGA, the fault injector operated at very high speeds.

Cieslewski et al. [9] used JTAG to improve fault injector portability with their Simple Portable FPGA Fault Injector (SPFFI). They have also compensated for the speed bottleneck of JTAG by designing SPFFI to only fault inject bits that are representative of a region of interest and/or fault inject random locations. Similar to the FLIPPER, they probabilistically determine design sensitivity. Guzman-Miranda et al. [10] have designed their FT-UNSHADES2 fault injection platform to obtain high-speed fault injection and full coverage. They used a standard Xilinx motherboard: the ML-510 with a Virtex-5 (XC5VFX130T). They can test custom-made daughtercards, which interface with the motherboard via PCI-Express. To maximize fault injection speed, FT-UNSHADES2 utilizes the SelectMAP interface. Their test design can work with a significant 512 bits of virtual input/output ports.

Starting with Virtex-6, Spartan-6, and 7-Series Xilinx FPGAs, Xilinx has releases a proprietary IP core called the Soft Error Mitigation (SEM) Core. The SEM Core is instantiated with the user design and uses the ICAP to detect, correct, and classify soft errors in the configuration memory of an FPGA device [11, 12]. While these fault injectors vary in technologies and methods used, they all have offered invaluable insight into how FPGA designs can be protected from SEUs.

5.3 XRTC Virtex-5 Fault Injector (XRTC-V5FI)

In conjunction with the Xilinx Radiation Test Consortium (XRTC), we built an FPGA fault injection system for testing digital FPGA circuits. Our main objectives in building this system were to achieve high customization and full bitstream coverage at a high fault injection rate. Because it takes a long time to complete fault injection on a full bitstream, we had to have a fast fault injector to increase the number of experiments completed. Also, a highly customizable system lets us conduct a larger variety of experiments and try different methodologies.

5.3.1 Architecture

The XRTC-V5FI fault injector (Fig. 5.5) is built using the XRTC motherboard, a test FPGA daughtercard, a non-volatile programmable read-only memory (PROM) card, and a host computer. The XRTC motherboard is also commonly used as a test fixture for radiation beam testing for other research projects. The test design is placed on the design under test (DUT) FPGA which is on the test daughtercard. The daughtercard allows us to run tests for both commercial and space-grade Virtex-5 FPGAs.

The XRTC motherboard has two service FPGAs (shown in Fig. 5.6) called the Configuration Monitor (ConfigMon) and the Functional Monitor (FuncMon). For our fault injection application, the ConfigMon performs scrubbing and readback and is responsible for configuring the DUT (pulsing PROG) and performing fault injection on the DUT (via SelectMAP), and logging sensitivity data for download. The FuncMon provides clock and reset signals, controls the fault injection sequence, compares design outputs, and signals the ConfigMon when an error occurs. The FuncMon and ConfigMon communicate directly with each other using a 16-bit wide Common Interconnect Bus (CI-Bus). The test design data is held on a PROM card plugged directly into the motherboard. This card contains the DUT golden bitstream file and the DUT mask bitstream file. The mask file is used to differentiate between the configuration bits used for logic, shift register LUTs (SRLs), and LUTRAM inside of configurable logic blocks (CLBs). The ConfigMon reads test design data from PROM card for fast configuration. The host PC computer communicates with both the ConfigMon and the FuncMon service FPGAs using RS-232

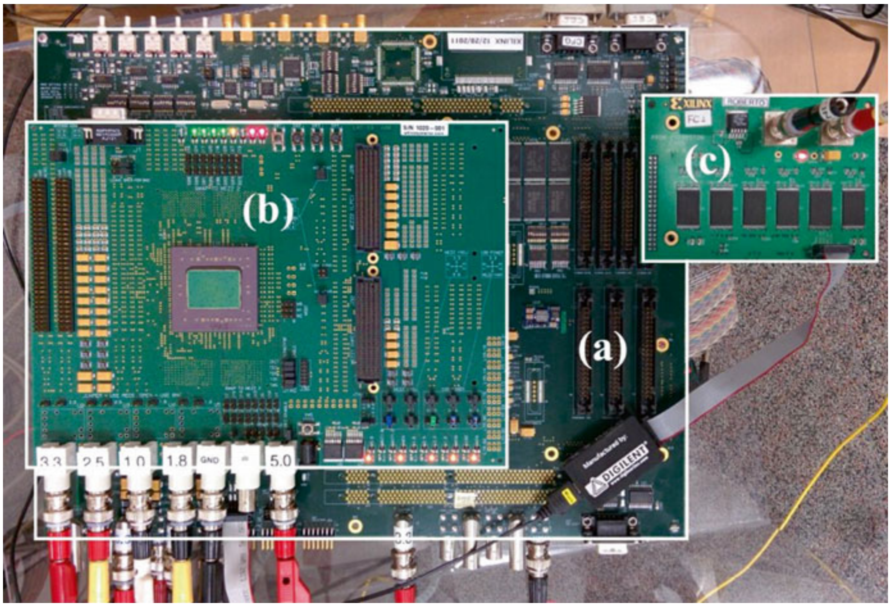

Fig. 5.5 Picture of (**a**) XRTC motherboard, (**b**) V5QV daughtercard, and (**c**) PROM memory card

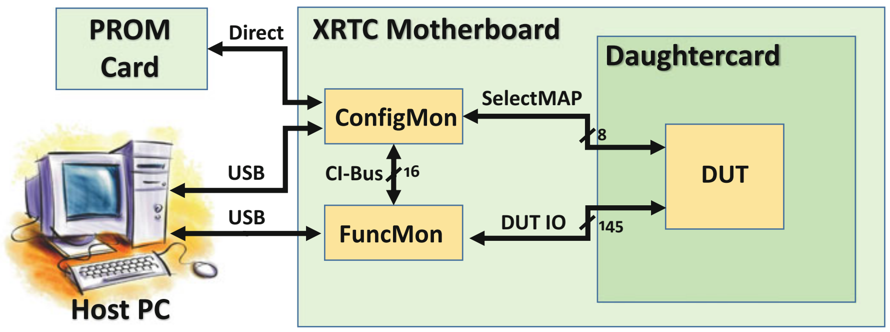

Fig. 5.6 High level view of XRTC-V5FI components

to initialize the system for fault injection, issue commands, and monitor the status and log data. The DUT FPGA receives its clock and reset signals from the FuncMon, and design outputs (145 signals) are sent from the DUT into the FuncMon for comparison. A high level illustration of the system is shown in Fig. 5.6.

The test FPGA for all the experiments described in this paper is the Virtex-5QV (V5QV). It is a 65-nm radiation-hardened by design (RHBD) FPGA manufactured by Xilinx, and it is qualified for space application [13]. The V5QV has 49,227,552 configuration bits, 34,087,072 of which are used for function and routing. There are also approximately 10.9 million bits used for block RAM (BRAM) and 4 million bits used for "testability and diagnostic reasons" [14]. For our experiments, we consider only the sensitivity of the bits used for function and routing.

5.3.2 Attributes

One major objective was to design our system to maximize fault injection speed. The current baseline time for a full bitstream fault injection campaign is 28 min. Design execution time and error recovery methods add additional time to the campaign. Each individual fault injection takes at least 49.1 µS. The ConfigMon configures and performs fault injection on the DUT via the SelectMAP port. The SelectMAP data port is 8-bits wide, and uses a 33 MHz clock. The XRTC-V5FI was designed to accurately measure configuration sensitivity by completely covering all 34.1 million configuration bits that control function and routing. The remaining 14.9 million bits in the bitstream are skipped.

Additionally, we have required that fault injection campaigns must be customizable. The FuncMon FPGA can be tailored for each design, allowing us to adjust the design execution time, test stimuli, fault injection procedure, and golden model. When comparing the design outputs, the FuncMon not only provides us with automatic error detection and recovery, but can also classify errors, determine faulty bit locations (e.g. a TMR voter error detection output), or other customizations based on the experiment. The host computer can request a snapshot of the faulty outputs if desired.

5.3.3 Methodology

Our experiments are built by placing two copies of the test design inside of the DUT FPGA. The outputs of each copy are assigned to 72 bits of the 145-bit signal that is outputted to the FuncMon. These outputs are then compared with each other at the end of a run cycle, and any mismatches are reported as errors. Alternatively, we could have had a golden model in the FuncMon and compared its outputs with a copy of the test design in the DUT, but we decided on the previous strategy to avoid any possible timing issues from comparing outputs from separate FPGAs.

Below is the fault injection loop procedure used for our experiments. This procedure is also shown with the diagram in Fig. 5.7.

1. The ConfigMon FPGA toggles the bit in the DUT FPGA's configuration memory.
2. The DUT is reset and its clock is enabled. The DUT is given time to load memories, execute software, and allow any errors to propagate through to its outputs.
3. The DUT's clock is stopped, and the outputs from both copies of the test design are compared with each other.
4. If an error is detected, the FuncMon signals the ConfigMon to record and log the error with the error's location and type.
   (a) For reset recovery experiments only, the configuration memory bit is restored and this process is repeated to determine if the error remained. The error is recorded as either recovered or unrecovered.
   (b) If a Single Event Functional Interrupt (SEFI) error (functional error independent of the test design) [15] is detected, the error is recorded, the DUT is fully reconfigured, and fault injection resumes at the next bit.

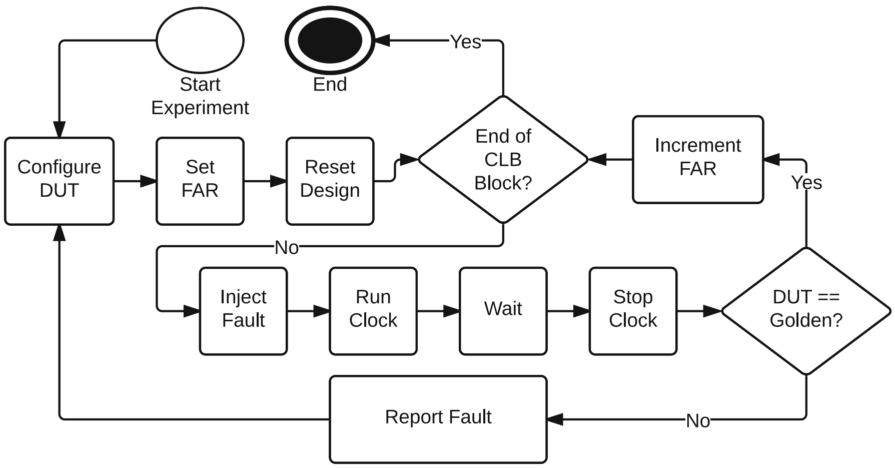

Fig. 5.7 Diagram showing the fault injection procedure

5. If the design contains a soft processor, we fully reconfigure the DUT after each detected output error to ensure the full recovery of memories.
6. The faulty bit is restored as the next bit is toggled.

At the beginning of a fault injection campaign, the host will reconfigure the service FPGAs and load the bitstream for the DUT FPGA onto the PROM card. The host will setup the ConfigMon with the correct parameters for fault injection, and test that the system is setup correctly. The host then commands the FuncMon to sequentially perform fault injection with a user-specified number of bits. The FuncMon will then run the fault injection procedure described above for each bit by issuing commands to the ConfigMon, waiting for the user-specified design execution time for each injected fault, and reporting results. The FuncMon reports the number of bits injected while the host ensures that errors are recovered and retrieves logged faults from the ConfigMon. The host keeps a database of errors with location and type, allowing for later analysis of the data.

5.4 Soft Processor Fault Injection

A soft processor is an implementation of a processor architecture that can be customized by the user for use on an FPGA. The key advantage soft processors offer to their users over standard microprocessors is the ability to optimize the hardware design for a particular application using FPGA resources. The reconfigurability of soft processors is also advantageous in that it allows the design to be updated whenever new features are desired, granting the processors relative immunity to obsolescence and enabling changes even when the FPGA has been deployed in a remote or harsh environment.

With a rise in the use of soft processors in harsh environments, a detailed understanding of soft processor reliability and failure modes is becoming indispensable. Using fault injection, we can test the configuration memory sensitivity of soft processors on FPGAs in an effort to understand their reliability and evaluate soft processor mitigation strategies and recovery methods. However, fault injection for soft processors involves grappling with a number of challenges unique to these designs. First, the reliability of a soft processor system depends not only on the specific hardware modules and features of the processor included in the system, but also on the software application the processor is executing. Since different software programs exercise a processor's functional units and memory in different ways, one software program may result in a different configuration memory sensitivity than another. A second challenge in soft processor fault injection is handling errors that propagate into memories. If an error from an injected fault propagates into a FPGA memory resource such as BRAM, LUTRAM, or an SRL, the error can persist in the memory even after a full system reset. Without special memory scrubbing or a full reconfiguration to repair the error, subsequent configuration bits may be deemed sensitive when a fault injection on a previous configuration bit was the real cause of the error. A third challenge in conducting fault injection experiments on soft processors is choosing a design runtime long enough to ensure that any bootloader code has completed and the desired software application is executing while also choosing a runtime short enough to minimize overall test time.

5.4.1 Soft Processors Used

For our fault injection experiments, we have used two of the most popular soft processor models: the MicroBlaze soft processor from Xilinx [16] and the LEON3 soft processor from Aeroflex Gaisler [17]. These experiments were run using identical embedded software applications and similar soft processor configurations, although there are still significant differences between the processor architectures.

The MicroBlaze is a 32-bit reduced instruction set computer (RISC) soft processor proprietary to Xilinx, built and optimized for use solely on Xilinx FPGAs [16]. It has a full Harvard architecture with separate data and instruction memory buses. The MicroBlaze is highly customizable, and Xilinx has produced a large number of compatible IP modules and libraries to use with it.

The LEON3 is an open-source 32-bit RISC soft processor from Aeroflex Gaisler [18]. It is based on the SPARC V8 architecture and supports a variety of operating systems such as Linux, RTEMS, and VxWorks. A ROM peripheral provided with the processor is used to decompress an application program stored in the ROM and loads it into processor main memory when no debugger is used. The bootloader code which performs this function is generated automatically by the LEON3 software tools and is stored in the ROM along with the compressed application code. A fault-tolerant version of the LEON3, the LEON3-FT, is commercially available from Aeroflex Gaisler as well.

5.4.2 Soft Processor Test Designs

For both the MicroBlaze and the LEON3, version 13.2 of the Xilinx tool flow was used to generate a bitstream. A simple Towers of Hanoi C program was compiled and run on each platform. Neither processor used an operating system for this test. No FPUs, MMUs, debug modules, or caches were enabled. All program memory was stored in the standard BRAM peripherals that came with the IP libraries for each processor. The MicroBlaze used an 8 KB BRAM while the LEON3 used a 32 KB BRAM. The LEON3 also included an additional 15 KB ROM to hold its bootloader code and a compressed version of the Towers of Hanoi program, which is copied into the RAM on startup by the bootloader. Each design ran on a 50 MHz clock input (supplied by the FuncMon) and was given 16,921 clock cycles to load and execute code memory. For each experiment, full reconfiguration was used to

Table 5.1 Comparison of configuration features used for experiments

MicroBlaze	LEON3
Version 8.20.a	GRLIB Release 1.3.4-b4140
5 Stage Pipeline	7 Stage Pipeline
No Register Windows	8 Register Windows
32-bit Multiplier	32-bit Multiplier
No Divider	32-bit Divider
Barrel Shifter	No Barrel Shifter
Pattern Comparator	No Pattern Comparator
2 BRAMS	16 BRAMS
Data and instruction LMB buses	Single AHB Bus

recover from reported errors to restore memories. Table 5.1 highlights some of the differences between the two processor configurations.

The processor outputs selected for the comparison between the DUT and golden versions of each soft processor design were chosen from each processor's bus signals governing memory access. From these outputs, we can determine if the faults affect the processor state in terms of the executed instructions and the calculated results being saved to memory. This strategy does not cover all possible design errors and would need to be adjusted for designs that interact with peripherals or use very little data memory.

For the MicroBlaze design, we observe the lower 16 bits of the address and data lines for both the data memory (dlmb) and the instruction memory (ilmb). We also monitor the memory enable and write enable nets. For the LEON3, we observe similar signals within the AMBA High-Performance Bus (AHB) Master In (ahbmi) and Slave In (ahbsi) signals from the ahbmi signal we observe the full 32-bit read data line and a 2-bit transaction response signal coming in from the bus slaves. From the ahbsi signal we observe the full 32-bit write data line and the lowest 6 bits of the address line coming out from the processor, which is the bus master.

5.5 Test Results and Analysis

The soft processors are duplicated and placed on the DUT FPGA. Figure 5.8 shows the layout of the MicroBlaze and LEON3 designs that were generated using Xilinx FPGA Editor software. The LEON3 is a larger design, occupying 2.28× the number of slices that the MicroBlaze occupies. Experiments were conducted to test for raw sensitivity and reset-recoverability. Result data was analyzed to determine the normalized sensitivity of a design, to compare the sensitive bit set of the design with the essential bit set generated by the Xilinx tools, and to determine a design's configuration memory error rates.

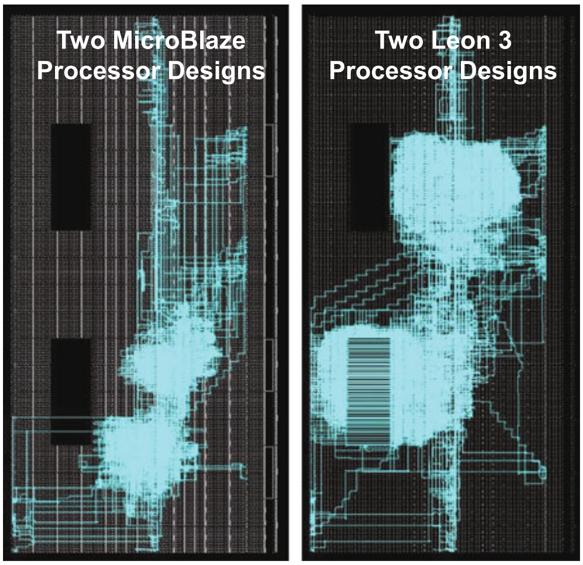
Two MicroBlaze Processor Designs
Two Leon 3 Processor Designs

Fig. 5.8 A layout for visual comparison of MicroBlaze and LEON3 designs (Generated with Xilinx FPGA Editor)

Table 5.2 Resource utilization

Design	Slices	Total LUTs	LUTs as logic	LUTs as RAM	Registers	BRAMs
MicroBlaze	1,029	2,493	2,190	128	1,601	4
LEON3	2,354	6,919	6,789	24	2,803	32

Table 5.3 Raw sensitivity results

Design	Sensitive bits	Sensitivity (%)
MicroBlaze	103,893	0.305
LEON3	188,378	0.553

5.5.1 Raw and Normalized Sensitivity

The raw sensitivity and resource utilization numbers for the MicroBlaze and LEON3 test cases are given in Tables 5.2 and 5.3. The LEON3 is both a larger design and had a larger number of sensitive bits than the MicroBlaze. The per-processor sensitivity is 51,946 errors for the MicroBlaze and 94,189 errors for the LEON3 design.

To compare the normalized design sensitivity, we use the following equation:

$$Normalized\ Sensitivity = \frac{Sensitivity}{Utilization} = \frac{(Total\ Slices)(Sensitive\ Bits)}{(Total\ Bits)(Used\ Slices)} \tag{5.1}$$

The normalized sensitivity results are listed in Table 5.4. The normalized sensitivity of the MicroBlaze is 26 % greater than the normalized sensitivity of the

Table 5.4 Total errors normalized over resources utilized

Design	Errors per slice	Errors per logic LUT	Errors per register	Normalized sensitivity (%)
MicroBlaze	100.97	47.44	64.89	6.07
LEON3	80.02	27.75	67.21	4.81

Table 5.5 Sensitive bits that were not recoverable by reset

Design	Sensitive bits	Unrecovered errors
MicroBlaze	104,001	14,271 (13.72 %)
LEON3	188,653	440 (0.28 %)

LEON3. We believe that the higher sensitivity of the MicroBlaze is due to how the two processors are made. The MicroBlaze, by default, is optimized for Xilinx FPGAs and uses LUTRAM and SRL primitives [16]. The LEON3 is for the most part FPGA architecture-independent, except for the primitives it uses to construct its Input/Output Blocks (IOBs), clock management devices, and memories, which are chosen through generics in its HDL code. Because more of the LEON3 design is synthesized than the MicroBlaze, this could result in less functional density and thus less sensitivity to upsets.

The static results from V5QV Single Event Effect (SEE) testing give an error rate of five static upsets per year for this FPGA's configuration memory [14]. Using this error rate, we would estimate a uniprocessor MicroBlaze design to have a mean time to configuration-induced failure (MTTCIF) of 131.24 years in GEO, and a small uniprocessor LEON3 design to have a MTTCIF of 72.38 years. It is important to keep in mind that this error rate does not include BRAMs or other user memories, and it does not account for Digital Clock Managers (DCMs), DSP48Es, Multi-Gigabit Transceivers (MGTs), and other non-CLB elements.

5.5.2 *Reset Recovery Experiment*

A system-wide reset can be a simple recovery technique for FPGA designs, however it does not always allow recovery of soft processor designs. When errors propagate into design memories, they can persist after a system reset. The goal of the reset-recovery experiment is to identify which configuration bits cannot be recovered. This experiment requires an additional step in our fault injection procedure where the fault-injected bit is corrected, the test design is reset, and the design outputs are again checked for errors. Table 5.5 shows how many unrecovered errors were found in each design. In the MicroBlaze design, about 1 in 7 sensitive bits were not recoverable by reset. In the LEON3, 1 in 429 were not recoverable. The LEON3 has a much better reset-recovery rate than the MicroBlaze design. We believe this is due to the bootstrap loader sequence that the LEON3 uses. When the reset is asserted, the LEON3 in effect scrubs its own program memory.

5.6 Conclusion

We have injected five billion bits over thousands of hours of testing to develop a unique Virtex-5 fault injection system. The fault injector was created with the XRTC motherboard and used to test the MicroBlaze and LEON3 soft-processors. The system performs fault injection successively on all configuration bits that control FPGA function and routing at a speed of 49.1 μS per bit. Our initial soft processor test results were shown, as well as processor reset recovery data. We found that the LEON3 has a lower normalized sensitivity and a higher reset-recovery rate than the MicroBlaze.

Future work with the fault injection system will focus on using the system to conduct experiments on soft processor designs. Fault injection experiments of the ARM Cortex-M0 and OpenRISC soft processors are underway, and other soft processors will be considered. In addition to performing experiments to determine the raw sensitivity of these processors, we will implement SEU mitigation and recovery techniques into the processor designs of the fault injection system and evaluate the effectiveness of each of these techniques in reducing design sensitivity. Using the data gathered from these tests, we will create reliability estimation tools and develop a model for estimating soft processor configuration sensitivity. These tests and tools will enable engineers to more fully understand the reliability tradeoffs in the use of soft processors, speeding up the design process, and allowing engineers to more accurately predict soft processor reliability in a variety of harsh environments.

Acknowledgments This work was supported by the I/UCRC Program of the National Science Foundation under Grant No. 1265957. We also acknowledge the Xilinx Radiation Test Consortium (XRTC) and members for support and use of XRTC test hardware.

References

1. Dodd PE, Massengill LW (2003) Basic mechanisms and modeling of single-event upset in digital microelectronics. IEEE Trans Nucl Sci 50(3):583–602
2. De Kastensmidt LFG, Neuberger G, Hentschke RF, Carro L, de Reis LRA (2002) Designing fault-tolerant techniques for SRAM-based FPGAs. IEEE Des Test Comput 21(6):552–562. doi:10.1109/MDT.2004.85
3. Mansour W, Velazco R (2013) An automated SEU fault-injection method and tool for HDL-based designs. IEEE Trans Nucl Sci 60(4):2728–2733. doi:10.1109/TNS.2013.2267097
4. Nazar G, Carro L (2012) Fast single-FPGA fault injection platform. In: Defect and fault tolerance in VLSI and nanotechnology systems (DFT), 2012 IEEE international symposium on, pp 152–157. doi:10.1109/DFT.2012.6378216
5. Lima F, Carmichael C, Fabula J, Padovani R, Reis R (2001) A fault injection analysis of Virtex FPGA TMR design methodology. In: 6th European conference on radiation and Its effects on components and systems, IEEE (2001), pp 275–282. doi:10.1109/RADECS.2001.1159293
6. Johnson E, Wirthlin MJ, Caffrey M (2002) Single-event upset simulation on an FPGA. In: Proceedings of the international conference on engineering of reconfigurable systems and algorithms (ERSA), CSREA Press, 2002, pp 68–73

7. Alderighi M, Casini F, d'Angelo S, Mancini M, Pastore S, Sechi GR (2007) Evaluation of single event upset mitigation schemes for SRAM based FPGAs using the FLIPPER fault injection platform. In: Proceedings of the 22nd IEEE international symposium on defect and fault-tolerance in VLSI systems, DFT'07, IEEE Computer Society, Washington, DC, USA, pp 105–113. doi:10.1109/DFT.2007.45
8. Sterpone L, Violante M (2007) A new partial reconfiguration-based fault-injection system to evaluate SEU effects in SRAM-based FPGAs. IEEE Trans Nucl Sci 54(4):965–970. doi:10.1109/TNS.2007.904080
9. Cieslewski G, George AD (2009) SPFFI: Simple portable FPGA fault injector. In: Proceedings of military and aerospace programmable logic devices conference (MAPLD)
10. Guzmán-Miranda H, Nápoles J, Mogollón J, Barrientos J, Sanz L, Aguirre M (2012) FT-UNSHADES2: a platform for early evaluation of ASIC and FPGA dependability using partial reconfiguration. La Sociedad de Arquitectura y Tecnologa de Computadores
11. LogiCORE IP soft error mitigation controller (2011) UG764 (v2.1)
12. Schumacher P (2012) SEU emulation environment. WP414 (v1.0)
13. Wang Y (2011) Recommendations for managing the configuration of the RHBD Virtex-5QV. In: Proceedings of military and aerospace programmable logic devices (MAPLD)
14. Swift G, Carmichael C, Allen G, Madias G, Miller E, Monreal R et al (2011) Compendium of XRTC radiation results on all single-event effects observed in the Virtex-5QV. In: Proceedings of NASA military and aerospace programmable logic devices (MAPLD)
15. White D (2011) Considerations surrounding single event effects in FPGAs, ASICs, and processors. WP402 (v1.0.1)
16. MicroBlaze Processor Reference Guide, Embedded Development Kit EDK 13.2 (2011). UG081 (v13.2)
17. Aeroflex gaisler LEON3 processor. http://www.gaisler.com/index.php/products/processors/leon3
18. Learn MW (2011) Evaluation of soft-core processors on a Xilinx Virtex-5 field programmable gate array. Sandia National Laboratories, Sandia Report No. SAND2011-2733, Apr 2011