Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 38 (2015) 277 – 282

www.elsevier.com/locate/procedia

The Fourth International Conference on Through-life Engineering Services

Modelling electronic circuit failures using a Xilinx FPGA system

Thomas E. Carney. Richard P. McWilliam. Alan Purvis.*

School of Engineering and Computing Sciences, University of Durham,

* Corresponding author. Tel.: +44 0771 3505409; *E-mail address:* alan.purvis@durham.ac.uk

Abstract

FPGAs are a ubiquitous electronic component utilised in a wide range of electronic systems across many industries. Almost all modern FPGAs employ SRAM based configuration memory elements which are susceptible to radiation induced soft errors. In high altitude and space applications, as well as in the nuclear and defence industries, such circuits must operate reliably in radiation-rich environments. A range of soft error mitigation techniques have been proposed but testing and qualification of new fault tolerant circuits can be an expensive and time consuming process. A novel method for simulating radiation-induced soft errors is presented that operates entirely within a laboratory environment and requires no hazardous exposure to radiation or expensive airborne test rigs. A system utilising modular redundancy is then implemented and tested under the new method. The test system is further demonstrated in conjunction with a software flight simulator to test single electronic modules in the context of active service on board a passenger aircraft and the effects of failure under radiation induced soft errors are observed. Our research proposes a test regime in which design strategies for self-healing circuits can be compared and demonstrated to work.

Keywords: FPGA; Hardware-in-loop; Fault Tolerance; TMR; Fault Injection;

1. Introduction

The Field Programmable Gate Array is a programmable logic device which has become ubiquitous in electronic systems due to its cost, ease of use and reprogrammability. The international market for FPGAs is huge and growing. The FPGA producer, Xilinx, reported 4th quarter sales in excess of half a billion US dollars in 2013 [1] while competitor, Lattice Semiconductor, have sold over one billion FPGAs over the 10 year period up to 2013, and are currently selling over a million units per day of their ultra-low density iCE FPGAs [2].

Despite their many advantages, FPGAs are susceptible to faults. Wirthlin et al. [3] provides a discussion of radiation induced faults on SRAM based FPGAs. These faults can take the form of (i) Total Ionizing Dose, causing permanent and irreparable damage to the device. (ii) Single Event Latchup (SEL) A potentially destructive condition in which a charged particle causes latchup within a device. Once in latchup, high currents will flow through parasitic bipolar transistors and destroy the device. (iii) Single Event Upset (SEU), a change in state of a digital memory element caused by an ionizing particle. These last are soft errors which do not cause any permanent damage to the device. [3] notes that devices such as the Xilinx QPRO high reliability FPGA are immune to latchup and has an acceptable dose tolerance, but are still sensitive to SEUs.

Anti-fuse FPGAs are available which are not susceptible to SEUs, but as noted in [4] these have their own drawbacks. They are one time programmable devices and are only available in smaller devices with a much smaller gate count than SRAM based versions.

Gusmao and Katensmidt [5] note the sensitivity of FPGAs to radiation while Niknahad et al. [6] point out that it is the SRAM memory elements inherent to most FPGAs which are particularly susceptible to SEUs (Single Event Upsets). Such errors may not present in systems in stable environments, and it is possible that some low priority systems may allow for such errors, but FPGAs have now become commonplace in critical applications in which transient faults cannot be tolerated. Such applications are pervasive in hazardous or remote applications including aircraft and spacecraft, where errors due to radiation are more likely.

Aircraft and spacecraft are prime examples of environments where these faults from radiation are likely to occur and absolutely cannot be tolerated.

[7] notes the effects of mechanical and/or ionizing stress during aircraft and spacecraft launch and operations, and the high incidence of cosmic rays and high energy photons encountered by FPGA systems operating in deep space.

doi:10.1016/j.procir.2015.08.039

It is clear then that fault tolerance and fault recovery in FPGAs is an absolutely vital area of research for aerospace applications. However, the testing and qualification of such systems for flight is of equal importance, and can be an extremely expensive process. [8] notes the expense and complexity involved in testing a piece of equipment under realistic operating conditions

1.1. TMR

TMR (Triple Modular Redundancy) is a technique commonly used in high radiation environments to avoid errors. It is the most common scheme for mitigating SEU (Single Event Upset) errors in sequential circuits in orbit. [9] TMR uses hardware redundancy to mask any single design failure by voting on the result of three identical copies of the same functional unit. [10]. Morgan et al. [11] points out several successful demonstrations of TMR implementation, but point out the drawbacks of the scheme in terms of the additional resources required. In fact TMR can require up to six times the area of the original circuit [3]

1.2. Configuration Scrubbing

Configuration scrubbing is the process of reloading the bitstream into configuration memory during operation. This can be carried out constantly or in the instance of fault detection. Adell and Allen [4] point out that this process will require an external, radiation hardened ASIC or one time programmable, Anti-Fuse FPGA. [4] also notes that scrubbing in itself is not a useful technique, but rather must be associated with a redundancy scheme in order to observe any improvement in reliability. This recommendation is echoed by Bridgeford et al. [12] as discussed in the section on Xilinx tools, below.

1.3. Xilinx Tools

Bridgeford Et Al [12] in the Xilinx application note ”Single Event Upset Mitigation Selection Guide” present a number of fault mitigation techniques which can be easily implemented through Xilinx' own software tools. The most basic approach noted is termed ”Power Cycling” i.e. switching the power off and then on again to force the FPGA to reconfigure.

1.4. Real World Effects

The failure rate of FPGA devices due to SEUs is entirely dependent on the level of radiation, most significantly neutron flux, and device cross section. [13] The largest Virtex-6 device (XC6VLX760) contains 184,823,072 bits of configuration memory and will exhibit a nominal failures-in-time (FIT) rate of 176 at sea-level in New York. While this represents a mean time between failures (MTBF) of 648 years, a system comprised of 1,000 FPGAs would experience a failure every year. [14] At high altitude (40000 feet), these numbers severely increase, namely as much as 100 times at the Equator and 560 times at the Earth poles [15].

Real world failures due to SEUs have been known to occur and pose a grave threat to the aviation industry. On 7 October 2008, Qantas flight 72 suffered a SEU in one of its three Air Data Inertial Reference Units causing the aircraft to carry out a sudden pitch down manoeuvre causing over 100 injuries to passengers including 14 serious injuries requiring passengers to be airlifted to hospital following an emergency landing. [16]

1.5. Testing Apparatus

Experiments for this investigation were carried out using the Xilinx ML605 evaluation board, featuring a Virtex-6 XC6VLX240T FPGA. Designs were developed in VHDL using ISE 14.7 and Xilinx System Generator in conjunction with Matlab 2013b. Though the problem of enhancing resilience has led to many design strategies there is a need to develop a standard test bed to stress these different circuits in repeatable, near real world engineering systems.

2. Error Injection System

The fault mitigation schemes put forward in this study are intended to be used in the presence of ionizing radiation, especially at high altitude, but it would be expensive, time consuming and potentially dangerous to test these schemes in such an environment and so it was necessary to develop a technique for simulating the effects of ionizing radiation by artificially introducing SEUs in the laboratory. One possible approach to introducing SEUs is Beam Testing. The device under test can be placed in a test chamber and bombarded with a beam of artificially generated radiation. Advantages to this technique are that the level of radiation can be accurately monitored and controlled, and that this is the closest possible simulation to actually exposing the hardware to natural ionising radiation. The obvious disadvantage to this approach is the immense cost associated with the development of a beam testing facility. [17], [18] and [19] all present details of such test schemes where purpose built facilities have been constructed. [17] and [18] present neutron beam testing facilities at different energies while [19] employs proton beam testing via a synchrotron device. All of these facilities represent a considerable investment of time and capital resources whilst only managing to perform testing over a small range of possible radiation sources.

If beam testing is considered to be too costly, then a simulation of SEU by local hardware is an acceptable alternative. [6] notes that the nature of SEUs is to invert SRAM memory elements within the FPGA. An external fault injection system could be developed using Xilinx Dynamic Reconfiguration to modify the state of the configuration memory. A method similar to this is presented in [20] where it is used to effectively simulate some SEU type errors. This approach is restricted by the limits of the Xilinx reconfiguration techniques which will actively prevent alteration of certain reserved bits in configuration memory, providing a level of protection which is not present in real-world applications.

The chosen solution was to use the Xilinx Soft Error Mitigation (SEM) IP Core. This core is included as part of the Xilinx CoreGen utility and is intended as an error detection and correction tool, but can also be configured to provide fault injection by inverting bits in the FPGA configuration memory via the

FPGA's Internal Configuration Access Port (ICAP). The error detection and correction capabilities of the SEM core are applicable only to a small selection of Xilinx' own FPGA chips. Furthermore, the design and operation of the SEM core is considered by Xilinx to be commercially sensitive information and is not available to form part of this study. For these reasons this will not be considered as a fault mitigation scheme for the purposes of this study. The Error Injection function is not required to be transferrable to other technologies and while its capabilities cannot be independently verified, they are fully documented in [21]. This makes the SEM core an appropriate and expedient tool to use for the implementation of error injection, although there are a number of issues which need to be overcome before the SEM IP core can be used in this manner.

2.1. Addressing

The SEM IP core is designed for testing the fault tolerance of an entire, chip-wide design, but the creation of such a design is a huge task and is well beyond the scope of this project. The Virtex-6 XC6VLX240T contains in excess of 57 million addressable bits of configuration memory of which no more than 6% was ever used during this investigation. The SEM core is capable of injecting errors into the configuration memory at an address specified at run time, but there is no direct mapping of configuration memory address space to FPGA fabric physical location of logic, in fact this mapping is deliberately withheld by Xilinx in order to protect the intellectual property of its customers. It is difficult therefore to be confident that errors are being injected into an address which will (i) compromise the design (ii) avoid compromising the SEM core itself. In order to determine a suitable address range for error injection, the following steps were taken:

1. Area Constraints
   The user constraint file included with the design was modified to place strict physical area constraints on the logic under test. This constraint restricts the CLB slice and Block RAM ranges where the logic can be placed as well as specifying that no other logic can be placed within this range.
2. FPGA Editor
   The FPGA editor program is a part of the ISE design tools. If run from the command line with the appropriate switches, this tool will automatically access the user constraint file mentioned above and incorporate it into the place and route process to generate the native circuit description and physical constraint files required to specify the physical implementation of the circuit. FPGA Editor can also create a log file which details how the area constraints in the user constraint file have been implemented.
3. Bitstream Generation
   The BitGen program uses the native circuit description and physical constraint files from the place and route process to create a bitstream to configure the device. If the proper command line switches are used, then BitGen is also capable of generating an essential bits file (.ebd). This file is a string of binary digits representing every bit in the bitstream, but rather than representing the actual data, a '1' represents a bit which is used in the design and a '0' represents an unused bit. Crucially, BitGen can be instructed to access the log file created by FPGA Editor and generate the essential bits file only for the parts of the design mentioned in the area constraints which were inserted into the user constraint file.
4. Essential Bit Analysis
   The .ebd file created by BitGen is still not particularly useful as it is in a proprietary format, details of which are not made available by Xilinx. Through negotiation with Xilinx it was possible to acquire the EBD Analyser — a piece of software which is not advertised or commercially available but is used by Xilinx engineers specifically for the purpose of analysing the EBD file and generating a list of linear frame addresses in configuration memory. Unfortunately the only version of the EBD Analyser in existence is designed exclusively for 7-series FPGAs. The SEM IP core for 7-series uses a slightly different address structure to the Virtex-6 version. Address structures are detailed in [21] and so it was possible to write a C++ program to parse the address file and generate a list of configuration memory addresses which were correctly structured for the Virtex-6.

2.2. Communication

The SEM IP core, when configured for error injection, has a 36 bit inject-address bus input and an inject-strobe input which can be used to inject errors to a known address. Unfortunately using these pins to inject errors to a series of non-sequential addresses while meeting timing requirements for injection proved impractical. Fortunately when the SEM IP core is instantiated it contains a PicoBlaze soft microcontroller. The PicoBlaze is a very small, very limited controller which in this case serves only as a control mechanism for the SEM IP core. Commands can be sent to, and data received from the pico-blaze using the monitor_tx and monitor_rx pins. In order to make use of this control mechanism, a soft RS232 communication module was implemented on the FPGA using the on-board UART of the ML605 board, and connected to the monitor_tx and monitor_rx pins of the controller. A proprietary software interface was written in the C# language to allow fault injection commands to be written to the on board UART from a windows PC.

2.3. Integration with System Generator

The above steps are sufficient to enable error injection via the SEM IP core into any design generated using the ISE tools, but in order to interface the design with the Flight Simulator in Simulink it was necessary to use the design in Hardware CoSimulation mode in System Generator. System Generator inherently takes control of all i/o on the FPGA which renders the USB communication mentioned above ineffective. To solve this problem it was necessary to redefine the ML605 board parameters within System Generator and reserve some pinouts as Non Memory Mapped (NMM) Ports. The NMM ports could be assigned to ports on the SEM IP block within system generator, effectively routing those i/o ports to the on-board resources.

3. N Modular Redundancy

The most common method of fault mitigation is Triple Modular Redundancy, or more generally, N-Modular Redundancy.

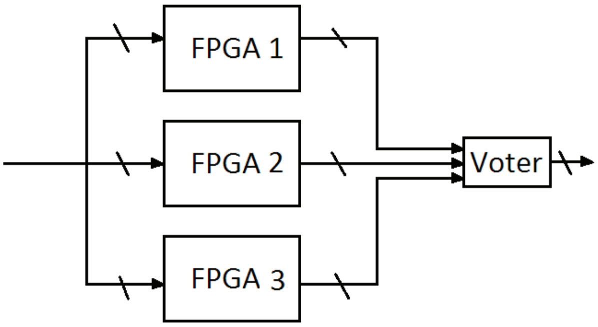

Fig. 1. Example of Triple Device Redundancy

The principle of operation of this scheme is to create multiple copies of the same module and apply a majority voting mechanism to the outcome. For this scheme to be effective, N must be greater than or equal to 3. Systems with N=2 may be able to detect an error but will not be able to correct it. The system of N-Modular redundancy which has been implemented here is similar to that presented by [22].

In implementing a system of modular redundancy it is necessary to decide the level at which redundancy will exist. The most reliable system would be device level redundancy. [22] notes that this method ”has the highest reliability for detecting single and multiple bit upsets, transient upsets, and any other functional interrupts including total device failure” [12] agree that ”duplicating the design on multiple FPGAs with voting on the outputs of the FPGAs provides the most robust mitigation scheme” This arrangement is shown in fig. 1. It is obvious however that this approach is the most costly in terms of resources, requiring multiple discrete devices as well as a radiation hardened voting device.

The Module level redundancy approach applies the same concept to the individual modules on the chip. Each module is replicated and a voting mechanism is applied for each output required. This level of redundancy provides a trade-off between resource utilisation and fault tolerance. It is possible to employ a more fine grained level of redundancy, with every individual component in the module, or even every logic gate or Slice LUT being replicated. This finest level of redundancy requires extremely high levels of resources, as discussed below.

The addition of the voting logic introduces a potential weakness into the system as this new component involves additional logic where errors may occur. It is difficult to envisage any manner in which a modular redundancy system may protect its own voting mechanism without introducing yet more unprotected voting mechanisms. This additional circuitry may lead to an increased likelihood of single faults occurring while simultaneously increasing the reliability of the overall system.

It is inherently difficult to implement n-modular redundancy for FPGAs due to the complex design flow involved in moving from HDL code to bitstream, most of which is beyond the control of the engineer. Synthesis tools such as Synopsis Synpliy and Xilinx Synthesis Tool (XST) perform optimisation operations on the schematics and HDL code created by the engineer and are explicitly designed to remove any redundant logic in order to maximise on-chip resource utilisation. To avoid this unwanted optimisation process it is necessary to use the design approach detailed in [22]. This includes designing all circuits using purely structural VHDL in order to gain an extra level of control over the synthesised design. It is also required to assign synthesiser attributes to the designed modules. These attributes vary between synthesis tools but when using XST it is necessary to apply the ”KEEP_HIERARCHY = true” attribute to the top level module in the design, and to apply the ”KEEP = true” and ”EQUIVALENT_REGISTER_REMOVAL = no” to all interconnecting signals.

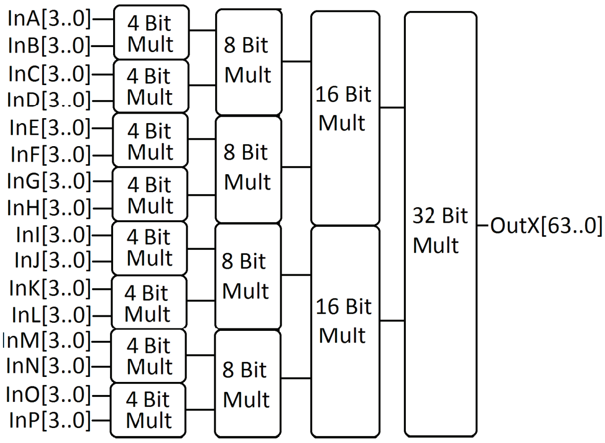

Fig. 2. Schematic diagram of the ripple multiplier circuit used to test redundant circuit

4. Device Under Test

For the first part of this study, the circuit being tested was an array of binary multipliers, with an input of 16 x 4bit numbers and outputting a single 64bit number. The numbers are multiplied together in stages, two at a time, with the outcome of each multiplication rippled forward to a next stage multiplier, as shown in fig 2. This configuration was chosen because it provides a simple, predictable calculation with a high number of bitwise logic functions. Each multiplier is composed of a shift register and ripple adder. These arithmetic components were built in structural VHDL at gate level in order to avoid being synthesised into any specialised arithmetic logic circuits on the FPGA, but rather to encourage synthesis into Slice LUTs in order to keep the results of the trial as generalised as possible.

Several versions of this circuit were implemented for analysis and testing as detailed below.

- Version 1 was unprotected and essentially identical to the version displayed in Fig 2. Following synthesis and placement this circuit was found to occupy 4245 Slice LUTs.
- Version 2 was designed with 3x module level redundancy and Version 3 was designed with 5x module level redundancy, as discussed in III above. Following synthesis and placement these circuits were found to occupy 17,564 and 29,138 Slice LUTs respectively.
- Version 4 was designed with triple component level redundancy. In this configuration each full adder, ripple

Table 1. Characteristics of the circuit designs created

Ver.	Replic. Count	Replic. Level	LUT Count	Res. Util.	Res. Increase
1	x1	NA	4245	2%	
2	x3	Module	17564	11%	x4.18
3	x5	Module	29138	19%	x6.86
4	x3	Component	176555	111%	

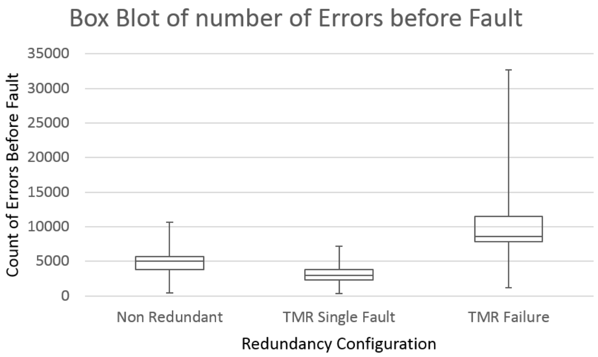

Fig. 3. Box plot showing the distribution of the number of errors injected before a fault is observed

adder and multiplier was triplicated with voting mechanisms added. Following synthesis and placement this circuit was found to occupy 176,555 Slice LUTs.

This data is summarised in table 1. It can be seen from the table that resource utilisation increases beyond the replication level, with each multiple of module level replication requiring approximately 1.375x increase in resources. Furthermore it should be noted that version 4 could not be studied further as it required more resources than were available on the Virtex-6 chip.

5. Testing and Results

Each circuit was implemented within the system generator hardware co-simulation model and subjected to a series of errors via the error injection system described above in section II. The inputs to the hardware were kept constant and consequently the output would normally be expected to remain constant. For circuit 1, a fault was considered to have occurred when the output from the circuit changed by one or more bits at which point the trial was ended and the system reset. For the TMR protected circuit 2 it was noted when a single one of the outputs changed, but the trial was continued until the final, voted output also changed.

The data collected from these trials is summarised in figs 3 and 4. and table 2. Note that the confidence interval stated in table 2 is adjusted for the fact that the number of errors before failure must be greater than or equal to 1.

It can be seen from fig 3 that the reliability of each individual output is reduced by the inclusion of module level redundancy, while the overall reliability of the system is improved. The total range of the data is greatly increased for the TMR circuit and there is considerable overlap between the three result sets, indicating that while the TMR circuit is likely to be more resilient to SEUs, it is possible that it may fail just as quickly. This is consistent with expectations given the random nature of the simulated SEUs which are equally likely to strike the final output voter as any other component.

Table 2. Summary of data

Version	Mean	Std Dev	99.7% interval
Non Redundant	4956.59	1988.2	1 — 10921
TMR Single Fault	3163.64	1530.08	1 — 7754
TMR Failure	9607.58	5921.41	1 — 27371

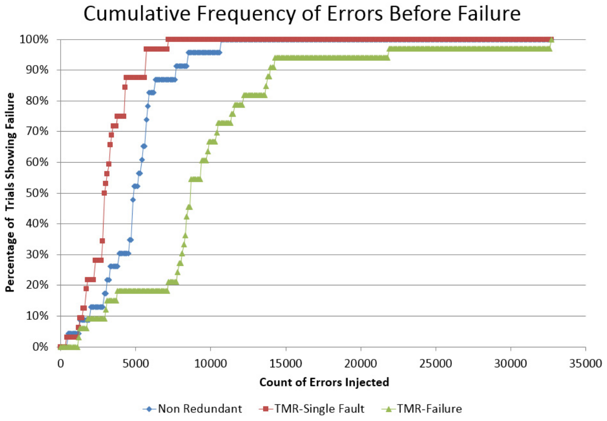

Fig. 4. Cumulative Frequency of the count of errors injected before a fault is observed

6. Flight Simulator Link

In order to assess the real-world effects of the error injection mechanism, as well as to achieve a realistic, full test environment of the circuits under test, the System Generator model was linked to the Flight Gear open source flight simulator. Flight gear contains some native support for interfacing with Matlab, via the Simulink Aerospace Blockset. This method of interface was used where possible due to its relative ease of use, but not all of the internal parameters used by Flight Gear can be accessed in this method, it was therefore necessary to implement a novel, more expandable interface. A schematic diagram of the complete test setup can be seen in fig.5.

FlightGear is capable of running with an inbuilt telnet server which can be used to access and set any values in its internal parameter tree via a set of telnet commands. In order to interface with the telnet server, a set of tcp sockets were set up within the Simulink model to act as i/o blocks. The Flight Simulator model used was an Airbus A380. A hardware model of the aircraft's pitot tube control electronics was developed. With the hardware-in-loop model running, the error injection application was used to introduce simulated SEUs into the FPGA SRAM memory in batches of 100 SEUs at a time.There were two failure modes present: Failure mode 1 was to cause the measured forward pressure to increase dramatically, by up to 30 orders of magnitude in some cases. This erroneous data caused the autopilot in the simulated aircraft to believe that its

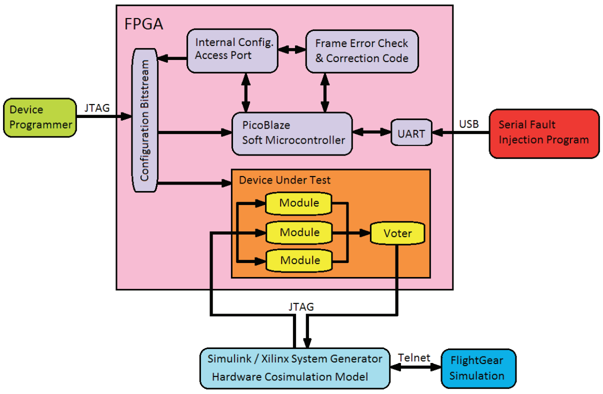
FPGA
Internal Config. Access Port
Frame Error Check & Correction Code
Configuration Bitstream
Device Programmer
JTAG
PicoBlaze Soft Microcontroller
UART
USB
Serial Fault Injection Program
Device Under Test
Module
Module
Module
Voter
JTAG
Simulink / Xilinx System Generator Hardware Cosimulation Model
Telnet
FlightGear Simulation

Fig. 5. Overall schematic diagram of the system, showing relevant circuitry within the FPGA as well as external components

speed was unnecessarily high and reduce the engine power to idle speed, causing the aircraft to rapidly lose altitude. Failure mode 2 was to cause the measured forward pressure to dramatically decrease, often to zero. This erroneous data would cause the autopilot to assume that its speed was dangerously low and throttle the engines to maximum in an attempt to compensate. Many devices can be modelled in hardware and placed in-loop in the flight simulator. SEU injection has been demonstrated but fault types such as stuck-at or 'no fault found' could be simulated, mitigation techniques implemented.

7. Conclusions

A Hardware-in-loop method for the qualification of FPGA circuits with radiation induced soft errors has been developed. It relies on generating random SEU type errors within the hardware under test without dangerous radiation or expensive airborne test rigs. The method can be used to assess modular redundancy in FPGA circuits and shows increased reliability for TMR circuits. Furthermore the method has been applied to the Flight Gear open source flight simulator to conduct hardware-in-loop testing of speed calculations from pressure readings and the resultant failure modes observed. It has been demonstrated that the fault injection tools developed to compare approaches to designing resilience into circuits work well as the TMR example shows.

Acknowledgements

This work was carried out with the support of the EPSRC Innovative Centre for Through-life Engineering Services. [EP/IO33246/1]. AP is also grateful for the support and welcome he received whilst a visitor at the University of Sydney Australian Centre for Field Robotics.

References

[1] "Xilinx investor information," http://investor.xilinx.com/releasedetail.cfm?ReleaseID=759113.

[2] EETimes, "With over 1 billion fpgas sold, lattice introduces machxo3 family," http://www.eetimes.com/document.asp?doc_id=1319597.

[3] M. Wirthlin, N. Rollins, M. Caffery, and P. Graham, "Hardness by design techniques for field programmable gate arrays," in *Proc. 11th annual NASA symposium on VLSI design*, Coeur d'Alene, Idaho, 28-29 May 2003.

[4] P. Adell and G. Allen, "Assessing and mitigating radiation effects in xilinx fpgas," *JPL Publ*, vol. 08-09, 2008.

[5] F. Gusmao, G. Kastensmidt, R. Hentschke, L. Carro, and R. Reis, "Designing fault tolerant techniques for sram-based fpgas," *IEEE Design and Test*, vol. 21, pp. 552–562, 2004.

[6] M. Niknahad, O. Sander, and J. Becker, "Qfdr — an integration of quadded logic for modern fpgas to tolerate high radiation effect rates," in *12th European Conference on Radiation and Its Effects on Components and Systems (RADECS)*, Sevilla, Spain, 19-23 Sept 2011, pp. 119–122.

[7] M. G. Parris, C. A. Sharma, and R. F. Demara, "Progress in autonomous fault recovery of field programmable gate arrays," *ACM Comput. Surv*, vol. 43, pp. 31:1–31:30, 2011.

[8] A. Griffio, D. Druryand, and D. Salt, "Hardware in the loop based synchronous generator emulation test rig for more electric aircraft power systems," in *6th IET International Conference on Power Electronics, Machines and Drives*, Bologna, Italy, 16-18 Oct 2012, pp. 1–6.

[9] S. Smith, "Single event upset mitigation by means of a sequential circuit freeze," *Microelectronics Reliability*, vol. 52, pp. 1233–1240, 2011.

[10] M. Straka, J. Kastil, Z. Kotasek, and L. Miculka, "Fault tolerant system design and seu injection based testing," *Microprocessors and Microsystems*, vol. 37, pp. 155–173, 2007.

[11] K. Morgan, D. McMurtrey, B. Pratt, and M. Wirthlin, "A comparison of tmr with alternative fault tolerant design techniques for fpgas," *IEEE Transactions on Nuclear Science*, vol. 54, pp. 2065–2072, 2007.

[12] B. Bridgeford, C. Carmichael, and C. W. Tseng, *Xilinx Application Note 987: Single-event upset mitigation selection guide*, 2008.

[13] C. Hu and Z. Suhail, *Xilinx Application Note 1073: NSEU Mitigation in Avionics Applications*, 2010.

[14] *Neutron Induced Single Event Upset FAQ*, Microsemi Corporation, 2011, www.microsemi.com/document-portal/doc_view/130760-neutron-seu-faq.

[15] C. Leong, J. Semiao, M. B. Santos, I. C. Teixeira, and J. P. Teixeira, "Seu sensor for short-term effects monitoring in fpgas," INESC-ID Technical Report, Tech. Rep., June 2013, http://www.inesc-id.pt/ficheiros/publicacoes/9174.pdf.

[16] A. T. S. Bureau, "In-flight upset 154km west of learmouth, wa, 7 october 2008," 2008.

[17] M. Osterlund, J. Blomgren, S. Pomp, and A. V. Prokofiev, "The uppsala neutron beam facility for electronics testing," *Nuclear Instruments and methods in Physics Research*, vol. B241, pp. 419–422, 2005.

[18] S. Lee, J. Rinckel, and P. E. Sokol, "New neutron radiation effects capabilities at the low energy neutron source (lens)," *Physics Procedia*, vol. 60, pp. 110–117, 2014.

[19] A. Hasanbegovic, "Proton beam characterization at oslo cyclotron laboratory for radiation testing of electronic devices," *16th Symposium on the Design and Diagnostics of Electronic Circuits and Systems*, pp. 135–140, 2013.

[20] M. Straka, J. Kastil, and Z. Kotasek, "Seu simulation framework for xilinx fpga: First step towards testing fault tolerant systems," *14th Euromicro conference on Digital System Design*, pp. 223–230, 2011.

[21] *Xilinx product guide PG036: LogiCORE IP Soft Error Mitigation Controller v4.0*, 2013.

[22] S. Habinc, "Functional triple modular redundancy: Vhdl design methodology," *Gaisler Research*, vol. v0.2, pp. 003–01, 2002.