

SINGLE EVENT UPSET DETECTION AND CORRECTION IN VIRTEX-4 AND VIRTEX-5 FPGAS



Bradley F. Dutton and Charles E. Stroud
Department of Electrical and Computer Engineering
Auburn University
Auburn, AL 36849-5201, USA
duttobf@auburn.edu strouce@auburn.edu

Abstract

A design for the detection and correction of single event upsets (SEUs) in the configuration memory of field programmable gate arrays (FPGAs) is presented. Larger configuration memories and shrinking design rules have caused concerns to rise about SEUs in high-reliability high-availability systems using FPGAs. We describe the operation and architecture of the proposed design as well as its implementation in Xilinx Virtex-4 and Virtex-5 FPGAs.¹

1 INTRODUCTION

The increased use of field programmable gate arrays (FPGAs) for implementing digital logic applications over the past two decades has been accompanied by increased concern about radiation effects, and, in particular, the effects of single event upsets (SEUs). Many FPGAs contain, in addition to memory elements such as flip-flops and random access memories (RAMs), a large static random access memory (SRAM) that establishes the overall application performed by the FPGA. An SEU induced bit-flip in the SRAM configuration memory, therefore, can alter the functionality of the FPGA. This makes SEUs of significantly more concern in FPGAs than in traditional application specific integrated circuits (ASICs). Radiation experiments indicate the SEU rate in FPGAs increased by a factor of 4.74:1 when design rules decreased from 600nm to 350nm with a corresponding reduction in power supply voltage from 5V to 3.3V [1]. On the other hand, Xilinx Virtex-4 FPGAs are reported to have SEU FIT (failures in 10⁹ hours) rates of 246 per 10⁶ bits of configuration memory, and only 151 per 10⁶ bits in Virtex-5 FPGAs [2]. This reduction in SEU FIT rate from Virtex-4 to Virtex-5 indicates that FPGA manufacturers are designing configuration memories to be more robust. However, the largest FPGAs currently have configuration memories with more than 80 million bits. Recent FPGAs, including Virtex-4 and Virtex-5, have incorporated mechanisms for the detection of SEUs in the configuration memory. Furthermore, these mechanisms can be used in conjunction with user-defined circuitry in the FPGA core to correct erroneous configuration memory bits that result from SEUs [3].

In this paper, we present an efficient SEU correction circuit that works in conjunction with existing SEU detection mechanisms in Virtex-4 and Virtex-5 FPGAs. This circuit can be synthesized and incorporated with user-defined digital applications in any Virtex-4 and Virtex-5 FPGA for detection and correction of SEUs during normal on-line system operation. We begin with an overview of existing SEU detection mechanisms in Section 2 along with an overview of previous work in SEU detection and correction in Virtex-4 FPGAs. The operation and architecture of the proposed SEU detection and correction circuit are presented Sections 3 and 4, respectively. Experimental results from the actual implementation of the proposed SEU detection and correction circuit in Virtex-4 and Virtex-5 FPGAs are presented in Sections 5 and 6 along with a comparison to previous work in Virtex-4 SEU detection and correction. The paper concludes with a summary in Section 7.

2 BACKGROUND

Like any other RAM, the configuration memory of an FPGA is partitioned into words, also called frames, which represent the smallest addressable unit of the memory for write and read operations. Virtex-4 and Virtex-5 frames consist of 1,312 bits [4][5]. Each frame includes a 12-bit field consisting of 11 Hamming bits and an overall parity bit for the frame data to provide the potential for single error correction (SEC) as well as double error detection (DED) in the frame data. The parity and Hamming bits are generated external to the FPGA by the configuration bitstream generation software and are subsequently downloaded with the application specific configuration data. However, system memory data subject to change during the operation of the FPGA, such as the contents of block RAMs and look-up tables (LUTs) used as distributed RAMs, are not covered by the parity and Hamming bits [6].

Virtex-4 and Virtex-5 provide a specialized core, called Frame ECC (error correcting code), for detection and identification of single and double-bit errors in the frame data [4][5]. For each frame read from the configuration memory, the Frame ECC module calculates the Hamming bits as well as the overall parity for the frame data, and compares these bits with the Hamming bits and parity for that frame stored in the configuration memory. Based on this comparison, the Frame ECC module produces indications for no error, single-bit error, and double-bit error conditions in addition to a syndrome

¹ This work was sponsored by the National Security Agency under contract H98230-04-C-1177 and supported in part by the National Science Foundation Grant CNS-0708962.

indicating the location of single-bit errors. The error codes for the Frame ECC primitive are shown in Table 1. System memory contents—block RAMs and LUT RAMs, for example—are masked from the internal parity and Hamming calculation by the Frame ECC.

Table 1. Frame ECC Error Codes [4][5]

Error Type	Condition (<i>syndromevalid</i> = 1)
No bit error	Hamming match, no parity error
1-bit correctable error (SEC)	Hamming mismatch, parity error
2-bit error detection (DED)	Hamming mismatch, no parity error

The Frame ECC function is performed each time a frame is read via a configuration interface such as the external serial Boundary Scan interface or parallel SelectMAP interface [4][5]. In addition, Virtex-4 and Virtex-5 include a 32-bit internal configuration access port (ICAP) that provides access to the configuration memory from within the FPGA core. The Frame ECC function is also performed each time a frame is read via the ICAP. Since the Frame ECC does not provide error correction, circuitry must be added in the FPGA fabric that uses the ICAP and Frame ECC modules to cycle through all frames and to detect and correct SEUs in the configuration memory.

An implementation of internal SEU detection and correction for Virtex-4 was reported in [3]. The design uses an 8-bit PicoBlaze [7] soft-core processor with additional circuitry and memory in the FPGA fabric for interfacing to the ICAP to read and write the configuration memory. The design can operate in a detection only mode, or can detect and correct single-bit errors. Because the block RAM used for the PicoBlaze program memory is not covered by Hamming bits, the basic design was later implemented in triple modular redundancy (TMR) [8]. However, both [3] and [8] are applicable only to Virtex-4; presently, no SEU detection and correction circuit for Virtex-5 has been reported.

3 OPERATION OF SEU DETECT AND CORRECT

The SEU detect and correct circuit, or *SEU controller* as it is referred to in this paper, is designed to be easily integrated into any existing VHDL-based user design with minimal overhead. At the top level, there are only two inputs—*clock* and *reset*—and one output—*error*. The VHDL component declaration for the SEU controller is presented in Figure 1.

```

component v5seu_controller is
generic (device: string(1 to 6));
port (rst, clock: in std_logic;
      error: out std_logic);
end component v5seu_controller;

```

Figure 1. SEU Controller Component Declaration

The generic *device* is a text string that specifies the device in which the SEU controller will be implemented. All Virtex-4 and Virtex-5 devices are supported. The *error* output is asserted when the first double-bit error is detected, and should trigger a reconfiguration of the FPGA from a reliable external memory, since multiple bit errors cannot be corrected by the SEU controller. The *clock* input is directly connected to the ICAP clock and the SEU controller. It is limited by the maximum ICAP clock frequency of 100 MHz, but can operate at any frequency below 100 MHz. The synchronous *reset* input forces the SEU controller into an inactive state, releasing the configuration interface for use by other applications. Asserting the *reset* input also resets the frame address to the first frame of configuration memory and clears the *error* output. When *reset* is released, the SEU controller will resume normal operation from the first frame of configuration memory on the next rising edge of *clock*. The *reset* input may be tied to logic 0 for free-running SEU detection and correction in user designs that do not require access to the configuration memory during normal operation. The operation of the SEU controller is as follows:

- 1 A 1312-bit frame of configuration memory is read through the ICAP as forty-one 32-bit words. The frame data is stored in a block RAM.
- 2 If an error is indicated by the outputs of the Frame ECC primitive, the type of error is determined as shown in Table 1. If the error indicates a double-bit error, the *error* output of the SEU controller is latched high and read back continues with the next frame of configuration memory. If a single-bit error is indicated, the location of the bit is determined from the syndrome and the erroneous bit is corrected (*i.e.* inverted) in the frame data stored in the block RAM.
- 3 The repaired frame is written back into the configuration memory at the same frame address from which it was read.
- 4 Read back resumes with the first frame of configuration memory in the configuration column containing the newly repaired frame.
- 5 When a configuration column has been completely read and repaired, the SEU controller advances to the next configuration column in the array.

4 SEU DETECT AND CORRECT ARCHITECTURE

The SEU controller is entirely implemented in configurable logic blocks (CLBs) and one block RAM in the FPGA fabric. It is constructed primarily around the ICAP and Frame ECC primitives [4][5]. The operation of the SEU Controller is managed with a finite state machine (FSM) implemented in logic slices. The FSM initiates reads and writes to the FPGA internal configuration memory and control registers via the 32-bit ICAP interface. A set of sixty-four 32-bit instructions are stored

in a 32×64-bit read-only memory (ROM) formed in 32 6-input LUTs in Virtex-5 and 128 4-input LUTs in Virtex-4. The 32×64-bit LUT ROM is addressed by a counter that is enabled or disabled by combinational logic from the FSM current state. The FSM also generates the frame address for reads and writes of the configuration memory. All reads from and writes to the configuration memory are 32-bits. The logic for the frame address counter is device dependent since every device has different numbers of rows and/or columns. Furthermore, the arrangement of different types of columns (*e.g.* CLB, DSP, RAM, etc.) can vary depending on the device. The generic *device* (see Figure 1) is used for determining and synthesizing the correct frame address logic for the given device.

The central component of the SEU controller architecture is a dual-port block RAM. (At least two columns of 18 kbit block RAMs are included in every Virtex-4 and Virtex-5 device). The block RAM is used to store each frame as it is read from the configuration memory. The A port of the block RAM is configured for 32-bit reads/writes, and the B port is configured for 1-bit reads/writes, as illustrated in Figure 2. The data inputs of the A port are wired directly to the outputs of the ICAP, and the A port data outputs are wired directly to the ICAP inputs via a 32-bit 2-to-1 multiplexer.

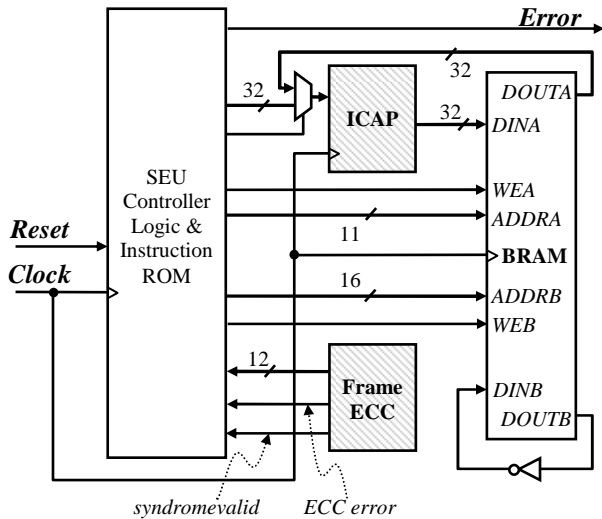


Figure 2. SEU Controller Block Diagram

The A port address inputs are controlled by a counter in the FSM. Every frame is read from the ICAP is stored in the first forty-one 32-bit words of the block RAM. Single-bit errors are corrected via the 1-bit B port interface. The B port address inputs are connected to combinational logic which provides the bit offset of the bit in error based on the Frame ECC *syndrome* outputs. The 1-bit B port data output is inverted in a LUT, and the output of the inverting LUT is connected to the 1-bit B port input. The B port write enable is controlled by

combinational logic from the *syndromevalid* and *ECC error* Frame ECC outputs in conjunction with the FSM. The location of single-bit errors within the frame is indicated by the *syndrome*[10:0] outputs of the Frame ECC primitive, however some additional combinational logic is required to determine the exact bit-offset of the error within the configuration frame. An efficient algorithm for determining the bit-offset of the error in the range 0-1311 is shown in Equation 1, where S[10:0] are the Frame ECC *syndrome* outputs [4][5].

$$\text{offset} = \{S[10:5] - 6'd22 - S[10], S[5:0]\} \quad (1)$$

If the binary value of *syndrome*[10:0] is 0 or a power of 2, then the error is located in one of the parity bits, in which case the location of the bit error is determined as shown in Table 2. The output of the syndrome combinational logic is tied to the B port address inputs. In this manner, the erroneous bit, as indicated by *syndrome*[11:0], is inverted when the block RAM B port write enable is asserted. The repaired frame is then written back into the configuration memory via the A port 32-bit output to the ICAP.

Table 2. Parity Bit Error Diagnosis [4][5]

<i>syndrome</i> [11:0]	offset	<i>syndrome</i> [11:0]	offset
100000000001	640	100001000000	646
100000000010	641	100010000000	647
100000000100	642	100100000000	648
100000001000	643	101000000000	649
100000010000	644	110000000000	650
100000100000	645	100000000000	651

A potential problem can arise when an odd number of multiple bit errors occur in a single frame of configuration memory. These errors will cause both a syndrome mismatch and overall parity mismatch, which collectively alias as a single-bit error (refer to Table 1). However, in this case, the syndrome outputs do not necessarily indicate the location of any of the actual errors, and can erroneously point anywhere in the range 0 to $2^{11}-1$ (2047). Since the actual frame data only exists in the range 0 to 1311, two scenarios are possible. In the first scenario, the odd-multiple bit error aliases as a single-bit error with the syndrome outputs pointing in the valid range of the frame data 0 to 1311. In response to the single-bit error indication, the SEU controller will invert the frame-bit pointed to by the syndrome, which will satisfy the Hamming code by creating a valid distance code word, and the modified frame will be written back into the configuration memory. The SEU controller will resume read back at the start of the configuration column containing the damaged frame. When the erroneous frame, now containing an even number of multiple errors, is read, the valid code word will cause a Hamming code match and an overall parity-bit match such that a “no bit error” indication is obtained. In the second scenario, when the frame containing an odd number of errors greater than one is read, the syndrome indicates an error

bit location in the range from 1312 to 2047. This range, while a valid address in the larger block RAM, lies outside of the range of valid frame data. Therefore, if events are allowed to proceed as in the first scenario, unmodified frame data would be written back into the configuration memory, effectively creating an infinite loop, since the same frame would be continually read from and written to the configuration memory without modification. Our solution, requiring minimal overhead, is to include a greater-than comparator in the SEU controller which detects when the syndrome points outside of the range of valid frame data (0 to 1311). When this condition occurs, the SEU controller ignores the syndrome and asserts the *error* output, indicating the existence of a multiple-bit error and that the FPGA configuration memory should be reloaded from a reliable external memory.

5 IMPLEMENTATION RESULTS

The greatest benefit of the SEU controller when compared to other approaches is the relatively high speed at which errors are detected and corrected. Figure 3 shows the time required for one full cycle of single-bit error correction and double-bit error detection in Virtex-4 devices for the Xilinx SEU controller [3] and our SEU controller, where a cycle is defined as the time to perform the operation over every configuration frame in the device, excluding frames containing block RAM contents.

The Xilinx SEU controller can operate in two modes: a single and double-bit error detection only mode, and a single-bit error correction and double-bit error detection mode [3]. As shown in Figure 3, the Xilinx “detect only” cycle time is nearly identical to our detect *and* correct mode. However, when single-bit error correction is enabled, the total cycle time for the Xilinx SEU controller increases to about 20 times that of our

normal detect *and* correct cycle time. On average, our SEU controller reduces the total cycle time for SEC and DED with respect to the Xilinx SEU controller by 94.7%. Figure 4 shows the total cycle time for the SEU controller in Virtex-5 devices. The cycle time is increased by an average of 17 μ s for each SEU detected and corrected. As can be seen in Figure 4, the repair time for one frame is negligible. However, the cycle time would double if there were one SEU present in every configuration column.

In addition to a 20 fold increase in speed for single-bit error detection and correction, our SEU controller is also less susceptible to corruption by SEUs. The Xilinx Virtex-4 SEU controller uses a PicoBlaze as its basic controller, which requires an 18 kbit block RAM for its program memory as well as LUT-RAMs for its internal scratchpad memory [7]. However, these memories are not covered by Hamming bits since the initials values can change during system operation such that an SEU in the program or scratchpad memories can affect the operation of the Xilinx SEU controller with the possibility of not detecting these SEUs and/or incorrectly modifying bits in the configuration memory as a result of SEUs in the program or scratchpad memory. The PicoBlaze program memory could be implemented in an ECC protected block RAM and the scratchpad memory could be expanded to include ECC protection, but this would require considerable changes in the PicoBlaze architecture, synthesis procedure, and supporting software (such as the assembly language compiler) as well as additional logic overhead. LUTs, on the other hand, are protected by Hamming bits. In our implementation of the SEU controller for Virtex-5, we intentionally implement the finite state machine (analogous to the PicoBlaze in [3]) in LUTs so that it will have some protection from SEUs with the possibility of detecting and correcting SEUs in the SEU controller logic. Therefore, our SEU controller is less susceptible to SEU induced failures.

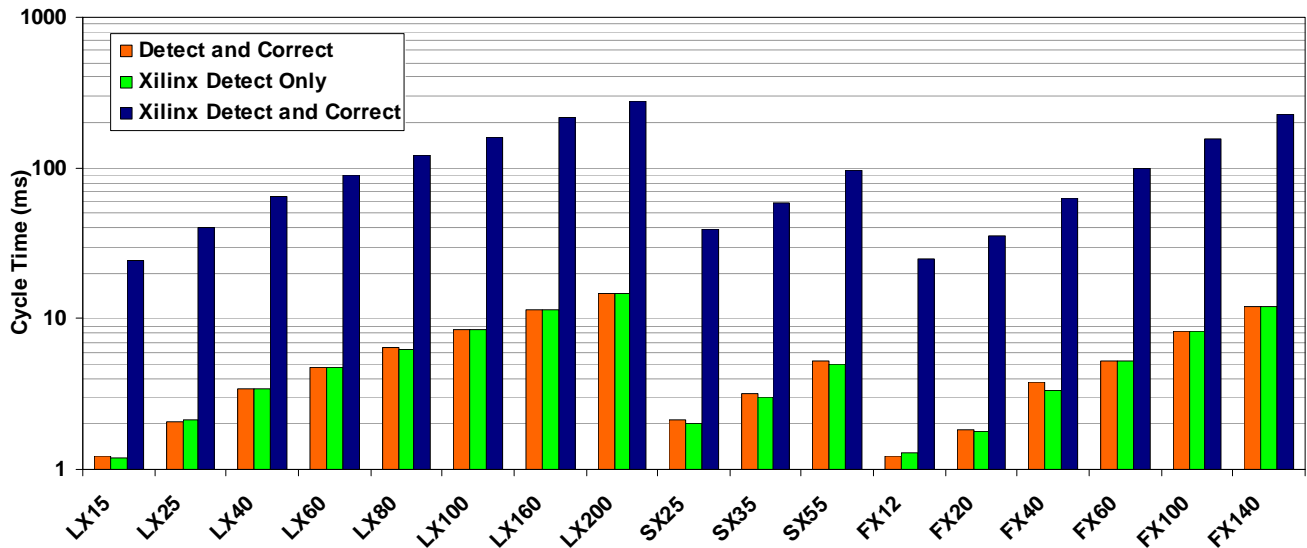


Figure 3. SEU Controller LOG Cycle Time vs. Virtex-4 Device

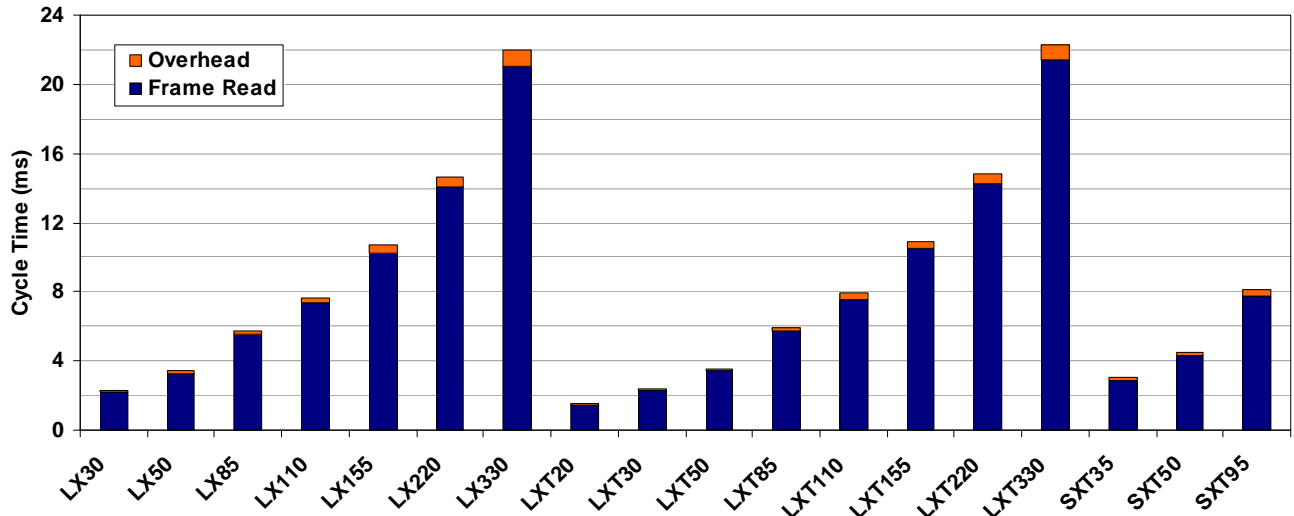


Figure 4. SEU Controller Cycle Time vs. Virtex-5 Device

In an effort increase the reliability of the Xilinx SEU controller, the authors of [8] triplicated the Xilinx Virtex-4 SEU controller and included majority voting circuits to implement TMR over all logic. TMR provides fault tolerance since an SEU affecting the operation of one of the three controllers is mitigated. However, this approach may be impractical for some applications because of its high overhead (in terms of total resource utilization). Additionally, an error affecting the PicoBlaze scratchpad memory or instruction block RAM still cannot be detected or corrected, since the bits associated with LUT RAMs are not covered by Hamming bits. A comparison of the device utilization for the Xilinx SEU controller [3], the Xilinx SEU controller with TMR [8], and our approach implemented in Virtex-4 is summarized in Table 3. While the Xilinx approach uses 23 fewer slices, we use one less block RAM and complete each cycle of the configuration an average of 20 times faster. Neither of the previous SEU controllers are directly applicable to Virtex-5 FPGAs, so a comparison of the resource utilization in Virtex-5 FPGAs is not possible.

Table 3. SEU Controller Resource Utilization

Resource	Xilinx [3]	Heiner <i>et al</i> [8]	Our SEU Controller
# Slices	149	1308	182
# Block RAMs	2	6	1
Avg Cycle (ms)	105.5	105.5	5.603
# Lines VHDL	3656	--	1051

The SEU controller has been synthesized for all Virtex-5 devices and has been downloaded and verified on LX30T, LX50T, SX35T, and SX50T. In addition, we have synthesized and verified the SEU controller in Virtex-4 FX12, SX35, and LX60 devices. The SEU controller requires one ICAP port, the Frame ECC port, one block RAM and about 60 logic slices in Virtex-5 devices (or one block RAM and about 182 slices in Virtex-4 devices). The number of utilized logic slices has

been observed to vary by ± 3 slices depending on the device and the area optimization of the place and route software tools in both Virtex-4 and Virtex-5 devices. During synthesis, the SEU controller logic and block RAM may be constrained to any area of the FPGA or may be left unconstrained for automatic placement. The power dissipation of our SEU controller measured on both Virtex-4 and Virtex-5 FPGAs is less than 5 mW at 100 MHz. Power requirements for the previous approaches in [3] and [8] were not reported.

6 EXPERIMENTAL RESULTS

For design verification, we developed an approach to emulate SEUs in the configuration memories of Virtex-5 FPGAs using a configuration memory read-modify-write process. The read-modify-write process is executed by an external 32-bit computer connected to the FPGA via the Boundary Scan configuration interface. A list of configuration bit addresses generated at random in software determines the locations for SEU injection. The SEU list generation software allows for control of the locations of the SEUs to an either specific region of or the entire configuration memory. Our approach is capable of injecting any number of errors in the configuration memory, simultaneously or individually, as determined by the length of the SEU target list.

The process begins by configuring the target device with the error-free SEU controller configuration. The SEU controller is held in reset while the SEUs are injected into the configuration memory. For each SEU in the list, the corresponding frame of configuration memory is read back from the target device to the external computer. The chosen bit in the frame is inverted, and the frame is written back to the same location in the configuration memory. After injection of the SEUs, the SEU controller is released from reset and allowed to run for one or more complete cycles. The number of single-

bit and multiple-bit errors are counted by internal counters included for analysis and verification only and these count values are read via the Boundary Scan interface at the end of the error correction/detection cycles. The success of the SEU controller is determined by comparing the values in the counters to the number of SEUs contained in the original list. Emulated configuration memory SEUs are classified in two categories. The first category includes all SEUs that are detected and corrected normally, as verified by the count values retrieved. The second category encompasses any SEU that affects the operation of the SEU controller such that either the SEU cannot be detected and corrected or the values contained in the counters are incorrect or cannot be retrieved for verification. A total of 8000 randomly generated SEUs were individually injected in the configuration memory of a Virtex-5 LX50T and the result of each trail was recorded. Our trials showed that, of the 8000 random SEUs, all but 178 were detected and corrected in the first full execution cycle, yielding a probability of detection and correction of 97.78%. Considering the SEU locations to be randomly distributed, independent samples, the lower bound for the probability of correction of SEUs at the 99% confidence level is 97.30% [9]. Therefore, the likely probability of detection and correction of any number of simultaneous SEUs greater than one is:

$$\text{Pr}(\text{correction}) = [1 - \text{Pr}(\text{failure})]^N \quad (2)$$

where N is the number of simultaneously occurring SEUs. The results of SEU emulation for 1000 SEUs in four Virtex-5 devices are shown in Table 4. In general, the percentage of correctable SEUs is positively correlated to the size of the configuration memory of the given device, since the number of configuration bits affecting the SEU controller functionality are fixed in relation to the total size of the configuration memory. For the largest Virtex-5 devices, greater than 99% probability of correction is expected. In our trials, 100% of SEUs that lie outside of the area of the configuration memory that controls the functionality of the SEU controller are corrected. The success rates for [3] and [8] were not reported.

Table 4. SEU Emulation Results

Device	Slice Count	Pop. Size (Mb)	Corrected/ Injected	Pr(correction) 99% Confidence
LX30T	59	7.29	950/1000	93.22%
SX35T	60	9.26	955/1000	93.46%
LX50T	59	10.9	980/1000	96.86%
SX50T	60	13.9	967/1000	94.96%
LX50T	59	10.9	7822/8000	97.35%

7 CONCLUSIONS

We have presented an SEU controller applicable to all Xilinx Virtex-4 and Virtex-5 FPGAs that is capable

of correcting single-bit errors and detecting double-bit errors in the FPGA configuration memory. The design is easily integrated in any existing user design with minimal resource overhead for detection and correction of single-bit errors using the ICAP. The SEU controller requires one block RAM and about 182 logic slices in Virtex-4, or one block RAM and about 60 slices in Virtex-5, and detects and corrects errors in the configuration memory 20 times faster than previously reported approaches in [3] and [8]. In addition, the design of our SEU controller makes it less susceptible to SEU induced failures by implementing the FSM and instruction ROM in LUT ROMs, which are covered by Hamming. However, SEUs that occur within the configuration bits that establish the SEU controller logic can, in some cases, cause the SEU controller to fail. The results of SEU emulation on hardware using the configuration memory read-modify-write approach showed that for all tested Virtex-5 devices, greater than 93% of SEUs are detected and corrected. This approach is shown in [10] to reproduce 97% of actual SEU induced faults in radiation testing.

8 REFERENCES

- [1] M. Ohlsson, P. Dyreklev, and K. Johansson, "Neutron Single Event Upsets in SRAM-Based FPGAs," *Proc. IEEE Nuclear and Space Radiation Effects Conf.*, pp. 177-180, 1998.
- [2] A. Lesea, "Continuing Experiments of Atmospheric Neutron Effects on Deep Submicron Integrated Circuits," WP286 (v1.0), Xilinx, Inc. 2008.²
- [3] L. Jones, "Single Event Upset (SEU) Detection and Correction Using Virtex-4 Devices," XAPP714 (v 1.5), Xilinx Inc., 2007.²
- [4] ___, "Virtex-4 FPGA Configuration User Guide," UG071 (v1.1), Xilinx Inc., 2008.²
- [5] ___, "Virtex-5 FPGA Configuration User Guide," UG191 (v3.2), Xilinx Inc., 2008.²
- [6] C. Carmichael and C. Wei Tseng, "Correcting SEUs in Virtex-4 Platform FPGA Configuration Memory," XAPP988, (v1.0), Xilinx Inc., 2008.²
- [7] ___, "PicoBlaze 8-bit Embedded Microcontroller User Guide," UG129 (v1.1.2), Xilinx Inc., 2008.²
- [8] J. Heiner, N. Collins, and M. Wirthlin, "Fault Tolerant ICAP Controller for High-Reliable Internal Scrubbing," *IEEE Aerospace Conf.*, pp. 1-10, 2008.
- [9] J. Sauro and J.R. Lewis, "Estimating Completion Rates From Small Samples Using Binomial Confidence Intervals," *Proc. Human Factors and Ergonomics Society*, pp. 2100-2104, 2005.³
- [10] E. Johnson, M. Caffrey, P. Graham, N. Rollins, and M. Wirthlin, "Accelerator Validation of an FPGA SEU Simulator," *IEEE Trans. on Nuclear Science*, vol. 50, no. 6, 2003.

² available at www.xilinx.com

³ available at www.measuringusability.com/wald