An Analytical Approach for Soft Error Rate Estimation of SRAM-Based FPGAs

Ghazanfar Asadi and Mehdi B. Tahoori

Northeastern University, Dept. of ECE, Boston MA 02115
Email :{gasadi, mtahoori}@ece.neu.edu

ABSTRACT

FPGA-based designs are more susceptible to single-event upsets (SEUs) compared to ASIC designs. Soft error rate (SER) estimation is a crucial step in design of soft error tolerant schemes to balance reliability, performance, and cost of the system. Previous techniques on FPGA SER estimation are based on time-consuming fault injection and simulation methods. In this paper, we present an analytical approach to estimate the failure rate of designs mapped into FPGAs. The proposed approach does not require physical implementation. Experiments show that this technique is orders of magnitude faster than fault injection while very accurate. We also report on Mean Time To Manifest (MTTM) error for different used resources of FPGAs.

1. Introduction

Nowadays, FPGAs are widely utilized in many application domains such as industrial, spacecraft and embedded applications due to their high performance, no Non-Refundable-Engineering cost and fast Time-To-Market.

Although FPGAs provides the advantages of low-cost design and fast Time-To- Market, the importance of dependability issues limit their widespread use in mission- critical applications [17]. FPGAs are vulnerable to *Single Event Upsets* (SEUs) [20]. SEUs are induced by energized particles hitting the silicon device. An SEU with sufficient energy changes the logic state of the memory element, producing a *soft error*. One possible solution to this problem is to use *radiation-hardened* FPGA devices. These devices, however, are much more expensive than Commercial-Off-The-Shelf (COTS) FPGAs; thus when cost is a major issue, the COTS devices are affordable [22]. Moreover, radiation-hardened devices are few generations behind state-of-the-art COTS devices.

A particle hit on a configuration bit cause a permanent error in the mapped design. Conventional fault-tolerant techniques [12] can protect only user-bits but not configuration bits. The only applicable fault-tolerant mechanism to protect configuration bits is *Triple Modular Redundancy* (TMR) scheme in all used logic and routing resources[6] [17]. This technique can be included in the design to be implemented by the COTS FPGAs. However, this solution enforces high area and performance penalties. It may not be affordable to put redundancy for all components when power and area are important constraints. To achieve a high level of reliability, efficient approaches combine both hardware (spatial) and software (temporal) redundancies.

The first step in designing such schemes is to study the effect of soft errors at the system level and identify the most vulnerable components in the system. Using this analysis and based on the redundancy budget, these components are protected with the highest priority (hardware or software redundancy).

Previous work on soft error rate (SER) estimation is simulation-based, radiation-based, or a combination of both [2] [5] [7] [8] [9] [17] [22]. All these methods are based on *Fault Injection* (FI). Using this methodology, a limited number of error sites are targeted for fault injection. Several workloads are then run to measure the number of detected failures by comparing the results of each run to the clean run. These steps make FI studies both time-consuming and inaccurate since the analysis is based on statistics. Moreover, these studies cannot be used during design phases since they need physical implementation.

In this paper, an accurate SER estimation method for FPGA designs is presented. The presented method does not require physical implementation, i.e. only a synthesis tool and a software program are used. In this method, first we compute the netlist error rate (the error rates of all circuit nodes). Then, some efficient graph algorithms are utilized to traverse the circuit topology from the erroneous node to the circuit outputs to compute the error propagation probability. Based on the netlist error rate and the propagation probabilities, we compute the failure rate of the system (the probability of an error appearing at system outputs). We also report on *Mean Time To Manifest* (MTTM) error for different used FPGA resources (e.g., routing, look-up tables, and control/clocking).

The rest of this paper organized as follows. Section 2 explains the previous SER estimation techniques. Section 3 describes the error models of SRAM-based FPGAs. The failure rate estimation method is given in Section 4. In Sec-

tion 5, the failure rate estimation of FPGA designs is presented. Experimental results are given in Section 6. Finally, Section 7 concludes the paper.

2. Previous Work

Error propagation probability (EPP) is one of the main factors of SER estimation of the circuit nodes [19]. Previous SER estimation techniques use fault injection and random-vector simulation [2] [5] [7] [8] [9] [17] [22]. To compute the EPP of a node, several random vectors are applied to the circuit inputs. Then the system outputs are observed to calculate the probability that the erroneous value of that node is synthesized by the input values and is propagated to the outputs. Fault injection is done using either radiation-equipment or bitstream alteration. Radiation-based methods [5] [7] are very expensive and they are not commonly used. These methods are mainly used for device characterization, not SER of a particular mapped design.

The methods presented in [2], [8], [9], [17], and [22] compute the SER of an implemented design based on configuration bitstream alteration. The device is configured for every faulty bitstream, i.e., one configuration bit is flipped for each workload. Then, it is run several clock cycles with different input vectors to compare the results with the golden-run results. These approaches can be classified into two categories, as follows. In the first category [2] [22], results are gathered and compared in a host system. So, it takes too much time to do experiments for all possible faults. In the second category [8] [9] [17], one FPGA is used for faulty run and one or two other FPGA(s) are used for the golden run and comparison of results. The prototype board consists of two or three FPGAs. To implement and evaluate larger designs, a new prototype board consist of higher density FPGAs is required.

3. Background: FPGA Error Models

The effects of SEUs in digital circuits can be classified as a) *transient* and b) *permanent* errors. SEUs can cause transient errors in the combinational logic resources, which can be propagated and captured in flip-flips. Also, SEUs can directly make transient errors on memory elements and change the contents of memory caches, main memories, register files and flip-flops. These errors are called transient because they may be overwritten or corrected using error-detection-and-correction techniques. So, transient errors impacts the user-defined logic and flip-flops of the FPGA.

Moreover, SEUs can make permanent errors in a FPGA if they alter the contents of configuration bits. Note that these errors differ from those errors which damage the device (hard errors). In this case, the configuration bit (and also the mapped design) remains erroneous until correct configuration is re-downloaded into the FPGA. So, these permanent errors are recoverable. In the rest of this paper, when we refer to permanent errors, we mean recoverable permanent errors.

The study and analysis of transient errors has been well described in [1], [4], [11], and [15]. They investigate the circuit behavior by injecting faults into the simulated or emulated model of the design. The fault injection in these techniques implies the alteration of memory elements such as data-path registers and control-unit registers, as well as alteration of input, output or internal signals [22]. Consequently, the effect of SEUs in the presence of the errors can be straightforwardly studied using common simulation or emulation tools.

The study of permanent errors due to configuration alteration requires more complex analysis since the simple bit-flip fault model cannot be exploited. An SEU in the device configuration bits can modify the interconnect inside a CLB. It can also affect the routing signals between different CLBs. Moreover, an SEU may change the functionality of the logic part by affecting the look- up tables (LUT). This issue has been addressed in [7], [17], and [22].

To summarize, there are two memory resources in FPGAs, a) user bits, and b) configuration bits. An SEU on user bits cause transient error, and an SEU on configuration bits lead to permanent errors.

3.1. Transient errors

Transient errors do not alter SRAM configuration bits but they affect user-defined logic and flip-flops as follow:

A bit-flip on combinational part inside CLBs: An SEU affecting a combination part makes a transient error in logic gates. This can be propagated to the sequential part and make a bit-flip error. Fig. 1 illustrates how an SEU makes a bit-flip error in a flip-flop. It has been shown that in ASIC designs, combinational logic is much less susceptible to soft errors than are memory elements [16] [24]. This is due to combinational logic provides some natural resistance to soft errors including logical masking, electrical masking, and latch-window masking [24].

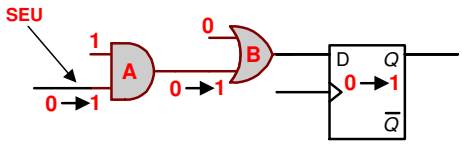

Fig. 1. An SEU affects one of inputs of the AND gate and makes a bit-flip error.

A bit-flip on user-defined flip-flops and memory elements: An SEU may directly affect the contents of flip-

Device	No. of FFs (Nff)	No. of Config. Bits (Ncb)	Nff/Ncb
XCV50	3,996	559,200	0.7%
XCV400	22,812	2,546,048	0.9%
XCV800	43,872	4,715,616	0.9%
XCV1000	56,832	6,127,744	0.9%

Table 1. Number of configuration bits versus number of flip-flops in Virtex FPGAs.

flops and memory elements. The flip- flop will remain erroneous until it is rewritten with another data or it is corrected by appropriate error detecting and correcting techniques.

3.2. Permanent errors

An SEU changing a configuration SRAM cell makes a permanent effect until the original configuration bitstream redownloaded into the FPGA. This type of error is the major error type in FPGAs because the number of SRAM cells dominates user-defined memory elements. Typically the number of SRAM configuration cells are more than 95% of all memory elements inside an FPGA [26]. As an example, Table 1 shows the number of configuration bits and the number of flip-flops for some Virtex FPGAs.

The permanent errors are classified similar to the classification given in [9]. They are routing error, LUT bit-flip, and control/clocking bit-flip.

Routing errors: Programmable interconnect points (PIPs), multiplexers and buffers constitute the programmable routing network of a Virtex FPGA. Typically, more than 80% of transistors in an FPGA are used in the routing network [25].

A routing signal may route *inter-CLBs* or *intra-CLB*. An inter-CLB routing signal connects two or more CLBs. Those that used inside a CLB are called intra-CLB signals. Switch matrices and line segments are used to route inter-CLB while multiplexers and buffers are mostly used for intra-CLBs. Select-bits of multiplexers comprise more than half of the susceptible SRAM cells to SEUs, as shown in [9].

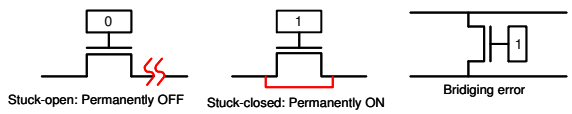

Fig. 2. An impact of SEU on routing signals

An SEU changing a configuration routing bit causes a switch open, switch short or bridging error (wired-or, wired-and), as shown in Fig. 2. Also, a PIP error may occur inside a switch matrix. Fig. 3 illustrate this situation. Consider a chain of PIPs connecting two nodes of the circuit. Any switch open in this chain results in an open error in the netlist. Also, if there is a switch short on an unused PIP connecting two nodes of the circuit, it results in a bridging error. Buffer errors are buffer on and buffer off. These are shown in Fig. 4.

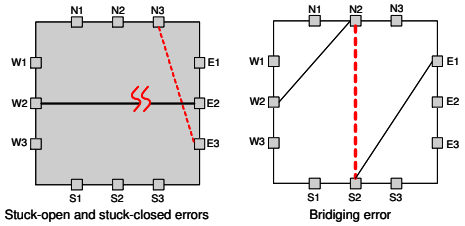

Fig. 3. A typical 3x3 switch matrix

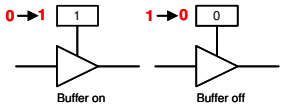

Fig. 4. Buffer error models

A bit-flip on LUT configuration bits: A look up table implement a logic function by storing all values for the truth table. Fig. 5 shows an example how a bit-flip changes the functionality of an LUT.

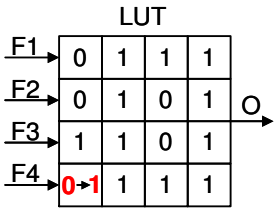

Fig. 5. LUT bit-flip

A bit-flip on control/clocking bits: There are control bits inside CLBs and IOBs to determine miscellaneous functionalities. As an example, there are some control bits that determine whether the LUT performs as a look up table, a dual-ported RAM, or a programmable shift register. Also, there are some SRAM cells to route clock signals throughout the circuit.

4. Failure Rate Estimation

Failure rate estimation is very important to a) assure the minimum reliability of the entire system, and b) to balance reliability, performance, and cost of the system. While particle flux uniformly encounters the entire system, the probability of an erroneous value being observed at the system

outputs highly depends on which node a particle is striked and the values of other nodes of the circuit at that time (system state). We recall that Error propagation probability (EPP) is one of the main factors of SER estimation of the circuit nodes [19]. SER estimation time using random-simulation exponentially increases with the size of the circuit. To overcome this difficulty, a fast and accurate technique to compute the EPP of a node is presented in [3]. We briefly describe the proposed method in the next subsection, then we adjust the proposed method for FPGA-based designs.

4.1. Error propagation probability computation

Here we use the signal probabilities of all nodes in the combinational part and then compute the error propagation probabilities based on the topological structure of the circuit. In the proposed approach, the structural paths from the error sites to all reachable outputs and flip-flops are extracted. Then, these paths are traversed to trace and compute the error propagation probability to reachable outputs or flip-flops. An *on-path* signal is a net on a path from the error site to a reachable output. Also, an *on-path gate* is defined as the gate with at least one on-path input. Finally, an *off-path* signal is a net that is not on-path and is an input of an on-path gate.

For error propagation probability calculation, as we traverse the paths, we use signal probability for off-path signals and use our *propagation rules* for on-path signals. The signal probability (SP) of a line l indicates the probability of l having logic value 1 [21]. SP techniques have been presented in [14, 18, 23]. The rules designed for on-path signals are mainly used to keep track of the polarity of propagated erroneous value to be used in reconvergent points. These rules are well described in [3].

Finally, we get *Netlist Failure Probability* vector ($N(n)$), where n is the total error sites. N_i is defined as the system failure probability given that node i is erroneous. Based on the experiments on ISCAS89 benchmarks, the accuracy of the presented approach (for transient errors) is within 6% of the random-simulation method while it is 4 to 5-order of magnitude faster.

5. Failure Rate Estimation of FPGA Designs

There are some differences between failure probability computation in ASIC designs and FPGA-based designs:

- In ASIC designs, we only need to compute the propagation of an erroneous value from the error site to POs or FFs. But in FPGA designs, we need to compute the activation probability as well as the propagation probability. This is because in FPGAs, failure occurs if the erroneous node is first activated and then propagated to POs or FFs (permanent errors).

- In FPGAs, the errors occurring in the configuration bits remain unchanged during the next clock cycles. So, the same failure probability is valid for the next clock cycles. however in ASIC designs, if an erroneous value, due to an SEU, is masked and not propagated to the outputs, the effect of the SEU will disappear in the next clock cycles.

- The error sites in ASICs are mainly logic gates rather than routing signals. But in FPGAs, routing signals (controlled by SRAM cells) constitute more than 70% of total sensitive SRAMs [9]. Sensitive SRAMs are those that any changes to them will eventually affect the user-bits.

- In FPGAs, if an SEU flips the content of a configuration bit, an erroneous value can propagate from the error site to the system outputs without any attenuation. But attenuation, also called *electrical masking* is one of the key factors that cause the combinational logic in ASIC designs to be much less susceptible to soft errors than memory elements [24]. Electrical masking occurs when the pulse resulting from a particle strike is attenuated by subsequent logic gates due to the electrical properties of the gates. This attenuation reaches to the point that the SEU does not affect system outputs.

To compute the failure rate of a design mapped into an FPGA, we perform the following steps. First, we compute the netlist failure probability of the system due to an error occurring on a node. Second, we compute the error rate of all nodes of the circuit. Third, the system failure is computed based on the previous steps. We are going into the details of each of these three steps in the following subsections.

5.1. Netlist Failure Probability (N_i)

Computation of N_i depends on the error model and the circuit topology. In general, N_i is the product of the activation probability of node i (AP_i) and the propagation probability of that node. We use signal probability of the node for AP_i. To compute the propagation probability, we use the method presented in Sec. 4..

In the case of open and stuck-at errors, N_i can be computed according to Equation 1. The first part of this equation accounts for the erroneous value being 0 and the second part accounts for the erroneous value being 1. Each part expresses that the erroneous value should be first activated (SP_i) and then propagated to the outputs (PP_i). Note that

the propagation probability of a signal depends only on the values of the other signals. So, $PP_i(0) = PP_i(1)$.

$$N_i = SP_i \times PP_i(0) + (1 - SP_i) \times PP_i(1) = PP_i \quad (1)$$

PP_i: *Propagation probability*
SP_i: *Signal probability of node i is used as activation probability*

In case of bridging wired-AND errors (between nets i and j), N_i can be computed according to Equation 2. The first part of the equation expresses the probability of node i being 1 and node j being 0. The second part computes the probability of node i being 0 and node j being 1. We use the same approach to compute N_i for bridging wired-OR errors between nets i and j, as shown in Equation 3.

$$\begin{aligned} N_i = &[SP_i \times (1 - SP_j) \times PP_i(0)] \\ &+[(1 - SP_i) \times SP_j \times PP_j(0)] \end{aligned} \quad (2)$$

$$\begin{aligned} N_i = &[SP_i \times (1 - SP_j) \times PP_j(1)] \\ &+[(1 - SP_i) \times SP_j \times PP_i(1)] \end{aligned} \quad (3)$$

In case of a bit-flip in one of LUT cells (cell i), N_i is computed as shown in Equation 4. To compute the failure probability due to LUT SRAMs, we use the propagation probability of the LUT output. It is computed as the product of the propagation probability of LUT output and the activation probability of the SRAM cell from LUT inputs (see Equation 4).

$$N_i = AP_i \times PP(LUT_{out}) \quad (4)$$

PP_i: *Propagation probability*
AP_i: *Activation probability of cell i computed during SP computation*

5.2. Node Error Rate (PR_i)

To compute the permanent error rates of the circuit nodes, $PR(n)$ is defined (n: total error sites), where PR_i is the permanent-error rate of node i. PR_i is calculated depending on the raw error rate of the device, the error model, and the number of SRAM configuration bits constituting the node. The error rate of a node of an FPGA design is directly proportional to the number of SRAM configuration cells controlling that node. So, nodes with more SRAM cells have higher error rates. For example, consider two nodes n_1 and n_2 consist of 4 and 2 pass-transistors (SRAM cells), respectively (Figure 6). The permanent-error probability of n_2 is less than n_1 because n_1 has more candidate locations for permanent-errors than n_2. PR_i is computed as shown in Equation 5. f is the total possible errors, which can occur on node i. For example, f equals to one for LUT SRAMs. f_{MUX} equals to the number of select bits. For a routing node, f is directly proportional to all ON/OFF switches connected to that node. As an example, the permanent-error rate of node AB, shown in Fig. 7, equals to six times the raw error rate of an SRAM-cell. The raw error rate of an SRAM cell (r) depends on the circuit device and the flux encounters the device. Current predictions show that typical FIT rates for latches and SRAM cells (measured at sea level) vary between 0.001-0.01 FIT/bit [10, 13, 20]. One FIT is equal to one failure in a billion hours (10^9). The FIT/bit also increases with elevation. At 10Km, the FIT/bit is approximately 100x higher [27].

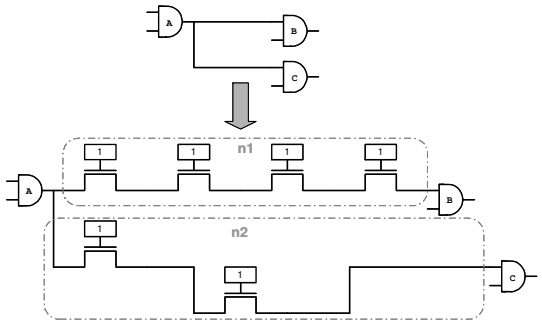

Fig. 6. Nodes n_1 and n_2 with different error rate

$$PR_i = r \times f_i \quad (5)$$

r: *Raw error rate of an SRAM cell*
f: *Number of all possible faults*

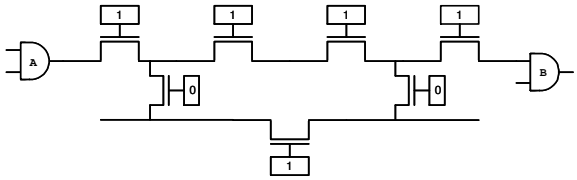

Fig. 7. Permanent-error rate of a routing node is directly proportional to the number of switches connected to the node

5.3. Failure rate computation

After computation of PR_i and N_i, the system failure probability due to node i can be compute as follows:

$$System\ failure\ rate\ S_i = PR_i \times N_i \tag{6}$$

Having the system failure rates computed for all nodes (S_i), we compute the system failure rate for the entire circuit in the clock cycle after the particle hit, according to the Equation 7.

$$S = 1 - \prod_{i=1}^{n} (1 - S_i) \tag{7}$$

Finally, the system failure rate of the whole circuit after c clock cycles from particle hit equals:

$$S = 1 - \prod_{i=1}^{n} (1 - PR_i \times (1 - N_i)^c) \tag{8}$$

6. Experimental Results

In the experiments, we use Xilinx FPGA devices [26]. To extract detail information (including used resources) of a design mapped into an FPGA, the Xilinx design language (XDL) is used. The error list considered in the experiments includes mux-open, PIP open, buffer-off, LUT bit-flip, and control/clocking bit-flip.

A software tool has been developed to extract the netlist information from the XDL file including the list of used resources, sensitive bits, and the error list. The failure rate of all circuit nodes are computed based on the above information. The experiments have been executed on a DELL PowerEdge 2650 © running Redhat Linux 9.0 © operating system.

We have mapped ISCAS89 benchmark circuits into Xilinx Virtex XCV300 devices. Table 2 shows the number of sensitive bits of the Virtex XCV300 device for ISCAS89 benchmark circuits. The sensitive bits are classified according to the error models described in Sec. 3. As shown in this table, the configuration routing bits constitute almost half of the total sensitive configuration bits. As shown in this table, the number of FFs, on average, is less than 0.5% of the number of configuration bits.

Table 3 shows the mean time to manifest errors from the error site to the system outputs. MTTM is classified for different types of configuration bits (routing, LUT, and control/clocking). The MTTM results have been measured for the Virtex XCV300 device. As shown in this table, the average MTTM of routing, LUT, and control/clocking resources for this device are 3.64, 25.63, and 1.63 cycles, respectively. This shows that control/clocking bits are the most sensitive ones. As the results show, the MTTM of the routing and control/clocking resources is much less than the MTTM of the LUT resources. These results also show that LUTs are the least sensitive bits to SEUs, although they are easiest to be protected against soft errors (implementation of parity schemes in LUTs is very straightforward). If we consider *normalized manifestation rate* for each category (routing, LUT, and control/clock) which is normalized to the number of sensitive bits as $\frac{\#sensitive\ bits}{MTTM}$, routing bits are the most vulnerable ones.

Circuit	Routing	LUT	Control/ Clocking	Total Conf Bits	No. of FFs
s298	445	410	133	988	14
s344	536	392	168	1096	15
s349	650	520	187	1357	15
s382	807	712	207	1726	21
s386	714	660	160	1534	6
s400	818	700	218	1736	21
s444	760	692	208	1660	21
s510	1328	1244	299	2871	6
s526	901	856	227	1984	21
s641	1418	1056	375	2849	19
s713	1320	988	355	2663	19
s953	2979	2644	597	6220	29
s1196	3320	2976	613	6909	18
s1238	3339	3224	652	7215	18
s1488	4013	3688	702	8403	6
s1494	4024	3628	695	8347	6
ave	1710	1524	362	3597	16

Table 2. Number of sensitive SRAM bits for each part

The detailed execution time of the presented SER estimation method is reported in Table 4. The total time of this SER estimation method includes a) the time needed to extract the netlist and also to extract SRAM cells information from the XDL file, b) SP computation time, and c) error propagation probability computation time. As shown in this table, the SER of an ISCAS89 circuit is computed, on average, in 38 seconds. As can be seen in this table, SP computation is the most time-consuming part of our estimation method.

The accuracy of our SER method versus the random-simulation method has also been reported in Table 4. The accuracy, on average, is about 95%.

The system failure rates of these benchmark circuits for the Virtex device are reported in Table 5. In these experiments, the raw error rate of an SRAM cell is assumed to be 0.01 (FIT/bit). Failure is observed for one and 50 clock cycles after an SEU flips the content of an SRAM cell. Since SEUs cause permanent errors in configuration bits, the failure rate in 50 cycles after an SEU occurrence is much higher than the failure rate in the first clock cycle.

Circuit	Routing	LUT	Cnt/clocking
s298	2.73	20.22	1.23
s344	2.59	17.48	1.37
s349	2.91	20.44	1.39
s382	3.30	22.11	1.40
s386	3.88	30.61	1.79
s400	3.13	20.80	1.40
s444	3.00	21.92	1.39
s510	4.87	34.77	2.14
s526	4.03	27.68	1.48
s641	2.42	16.96	1.41
s713	2.41	16.82	1.37
s953	3.23	21.22	1.49
s1196	5.17	36.59	2.16
s1238	5.56	41.23	2.33
s1488	4.47	29.92	1.90
s1494	4.60	31.39	1.98
ave	3.64	25.63	1.63

Table 3. Mean Time To Manifest (MTTM) errors to outputs (Results in terms of cycles)

Circuit	Execution Time (sec)				Accuracy (%)
	Extract	SP	SFR	Total	
s298	0.01	0.46	0.01	0.48	95
s344	0.01	12.83	0.01	12.85	96
s349	0.01	15.47	0.01	15.49	96
s382	0.01	17.93	0.01	17.95	95
s386	0.02	1.05	0.01	1.08	97
s400	0.02	17.66	0.01	17.69	95
s444	0.02	17.27	0.01	17.30	95
s510	0.03	31.36	0.01	31.40	96
s526	0.02	20.54	0.01	20.57	96
s641	0.04	32.27	0.06	32.37	91
s713	0.06	28.43	0.05	28.54	90
s953	0.14	65.10	0.10	65.24	95
s1196	0.19	76.50	0.21	76.90	96
s1238	0.21	83.41	0.25	83.87	96
s1488	0.22	95.07	0.13	95.42	95
s1494	0.19	95.65	0.14	95.98	95
ave	0.08	38.18	0.06	38.32	95

Table 4. System Failure Estimation Time.
SP Time: Signal Probability computation time (second)
SFR Time: System Failure Rate computation time (second)
Number of Clock cycles: 50

7. Conclusions

Designs mapped into FPGAs are more susceptible to soft errors than ASIC implementation. Analysis of the effect of soft errors in different resources of an FPGA is a key factor in development of low cost, high performance, and high reliable solutions.

In this paper, a very fast and accurate SER estimation technique for FPGA-based designs has been presented. The experimental results show that the presented method is orders of magnitude faster than conventional fault injection methods while more than 95% accurate. Moreover, the results show that the error manifestation time for routing and also clock/control resources are 10 times less than that for LUTs.

8. References

[1] L. Antoni, R. Leveugle, and B. Feher, "Using Run-Time Reconfiguration for Fault Injection in Hardware Prototypes," Proceedings of the IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT), pp. 405-413, 2000.

[2] G. Asadi, G. Miremadi, H.R. Zarandi, and A. Ejlali, "Fault Injection into SRAM-Based FPGAs for the Analysis of SEU Effects," Proceedings of the IEEE International Conference on Field-Programmable Technology (FPT), pp. 428-430, Tokyo, Japan, Dec 2003.

[3] G. Asadi and M. B. Tahoori, "An Accurate SER Estimation Method Based on Based on Propagation Probability," to appear in the Proc. of the Intl. Conference on Design, Automation and Test in Europe (DATE), 2005.

[4] J. Arlat, M. Aguera, L. Amat, Y. Crouzet, J.C. Fabre, J.-C. Laprie, E. Martins, and D. Powell, "Fault Injection for Dependability Validation: A Methodology and Some Applications," IEEE Transactions on Software Engineering, Vol. 16, No. 2, February 1990.

[5] C. Carmichael, E. Fuller, J. Fabula, and F. Lima, "Proton Testing of SEU Mitigation Methods for the Virtex FPGA," Proceedings of the Military and Aerospace Applications of Programmable Logic Devices (MAPLD), 2001.

[6] C. Carmichael, E. Fuller, P. Blain, and M. Caffrey, "SEU Mitigation Techniques for Virtex FPGAs in Space Applications," Proceedings of the Military and Aerospace Applications of Programmable Logic Devices (MAPLD), 1999.

[7] E. Fuller, M. Caffrey, A. Salazar, C. Carmichael, and J. Fabula, " Radiation Characterization and SEU Mitigation of the Virtex FPGA for Space-Based Reconfigurable Computing," Proc. of the IEEE Nuclear and Space Radiation Effects Conference (NSREC), Nevada, July 2000.

[8] M. Gokhale, P. Graham, E. Johnson, N. Rollins, and M. Wirthlin, "Dynamic Reconfiguration for Management of Radiation-Induced Faults in FPGAs," Proceedings of the 18^{th} International Parallel and Distributed Processing Symposium (IPDPS'04), pp. 145-150, Santa Fe, New Mexico, April 2004.

[9] P. Graham, M. Caffrey, J. Zimmerman, D. E. Johnson, P. Sundararajan, and C. Patterson, "Consequences and Categories of SRAM FPGA Configuration SEUs," Proceedings of the Military and Aerospace Applications of Programmable Logic Devices (MAPLD), Washington DC, September 2003.

[10] S. Hareland, J. Maiz, M. Alavi, K. Mistry, S. Walstra, and C. Dai, "Impact of CMOS Scaling and SOI on soft error rates of logic processes," Symposium on VLSI Technology, Digest of Technical Papers, PP. 73-74, June 2001.

Circuit	System Failure Rate	
	clk=1	clk=50
s298	2.90	8.28
s344	3.51	9.10
s349	3.82	11.2
s382	4.23	13.8
s386	2.94	9.30
s400	4.50	14.1
s444	4.33	13.6
s510	4.47	19.1
s526	4.06	13.6
s641	9.10	22.9
s713	8.64	21.4
s953	14.4	41.1
s1196	10.0	37.9
s1238	9.57	35.9
s1488	13.8	58.3
s1494	13.3	57.7
ave	7.09	24.20

Table 5. System Failure Rate of ISCAS89 circuits mapped into the Virtex XCV300 device.
FIT of an SRAM cell: 0.01

[11] R. K. Iyer and D. Tang, "Experimental Analysis of Computer System Dependability," in Chapter 5 of Fault-Tolerant Computer System Design, D. K. Pradhan (ed.), Prentice-Hall, 1996.

[12] B. W. Johnson, "Design & analysis of fault tolerant digital systems," Addison-Wesley Longman Publishing, ISBN:0-201-07570-9, Boston, MA, 1988.

[13] T. Karnik, B. Bloechel, K. Soumyanath, V. De, and S. Borkar, "Scaling Trends of Cosmic Rays Induced Soft Errors in Static Latches Beyond 0.18μ," Symposium on VLSI Circuits, Digest of Technical Papers, pp. 61-62, June 2001.

[14] R. Kodavarti and D. E. Ross, "Signal Probability Calculation Using Partial Functional Manipulation," Proc. of the VLSI Test Sym., [Digest of Papers], 1993.

[15] R. Leveugle, "Fault Injection in VHDL Descriptions and Emulation," Proceedings of the IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, pp. 414-419, JAPAN, October 2000.

[16] P. Liden, P. Dahlgren, R. Johansson, and J. Karlsson, "On Latching Probability of Particle Induced Transients in Combinational Networks," Proceedings of the 24^{th} Symposium on Fault-Tolerant Computing (FTCS-24), pp. 340-349, 1994.

[17] F. Lima, C. Carmichael, J. Fabula, R. Padovani, and R. Reis, "A Fault Injection Analysis of Virtex FPGA TMR Design Methodology," Proceedings of the Radiation Effects on Components and Systems Conference (RADECS2001), Grenoble, FRANCE, 2001.

[18] A. Majumdar and S. B. K. Vrudhula, "Analysis of Signal Probability in Logic Circuits Using Stochastic Models," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, pp. 365-379, Vol. 1, No. 3, Sep. 1993.

[19] K. Mohanram and N. A. Touba, "Cost-Effective Approach for Reducing Soft Error Failure Rate in Logic Circuits," Proceedings of the International Test Conference (ITC), pp. 893-901, Charlotte, NC, October 2003.

[20] E. Normand, "Single Event Upset at Ground Level," IEEE Transactions on Nuclear Science, vol. 43, No. 6, December 1996.

[21] K. P. Parker and E. J. McCluskey, "Probabilistic Treatment of General Combinational Networks," IEEE Trans. on Computers, Vol. c-24, No.6, pp. 668-670, June 1975.

[22] M. Rebaudengo, M. Sonza Reorda, and M. Violante, "Simulation-Based Analysis of SEU Effects on SRAM-Based FPGAs," Proceeding of the 12^{th} International Conference on Field-Programmable Logic and Applications (FPL2002), September 2002.

[23] J. Savir, G. S. Ditlow, and P. H. Bardell, "Random Pattern Testability," IEEE Trans. on Computers, Vol. c-33, No. 1, pp. 79-90, January 1984.

[24] P. Shivakumar, M. Kistler, S. W. Keckler, D. Burger, L. Alvisi, "Modeling the Effect of Technology Trends on the Soft Error Rate of Combinational Logic," Proc. of the International Conference on Dependable Systems and Networks (DSN'02), Washington D.C., June 2002.

[25] M. B. Tahoori, S. Mitra, S. Toutounchi, and E. J. McCluskey, "Fault Grading FPGA Interconnect Test Configurations," Proceedings of the International Test Conference (ITC), pp. 608-617, Oct. 2002.

[26] Xilinx, "Virtex 2.5V Field Programmable Gate Arrays," Data Sheet DS003-1, Xilinx, San Jose, CA, April 2001.

[27] J. F. Ziegler, "Terrestrial Cosmic Rays," IBM Journal of Research and Development, pp. 19-39, Vol. 40, No. 1, Janaury 1996.