Soft Error Susceptibility Analysis of SRAM-Based FPGAs in High-Performance Information Systems

Hossein Asadi, *Student Member, IEEE*, Mehdi B. Tahoori, *Member, IEEE*, Brian Mullins, *Student Member, IEEE*, David Kaeli, *Member, IEEE*, and Kevin Granlund

***Abstract*—Soft errors due to cosmic particles are a growing reliability threat for VLSI systems. The vulnerability of FPGA-based designs to soft errors is higher than ASIC implementations since the majority of chip real estate is dedicated to memory bits, configuration bits, and user bits. Moreover, single event upsets (SEUs) in the configuration bits of SRAM-based FPGAs result in permanent errors in the mapped design.**

In this paper we analyze the soft error vulnerability of FPGAs used in information systems. Since the reliability requirements of these high performance information subsystems are very stringent, the reliability of the FPGA chips used in the design of such systems plays a critical role in overall system reliability. We present an analytical approach (versus fault injection) for soft error rate estimation in FPGA-based designs. We also validate the projections produced by our analytical model using field error rates obtained from failure data obtained from a large FPGA-based design used in the Logical Unit Module board of a commercial information system. This comparison confirms that the projections obtained from our analytical tool are accurate (there is an 81% overlap in FIT rate range obtained with our analytical modeling framework and the field failure data studied).

***Index Terms*—Error analysis, field programmable gate arrays (FPGA), information systems, logic circuit fault tolerance, reliability estimation, reliability modeling, SRAM chips (SRAM-based FPGAs).**

I. Introduction

SOFT errors are intermittent malfunctions of hardware that are not reproducible [29]. These errors, also called transient errors, occur more often than permanent errors [17]. *Single Event Upsets* (SEUs) that cause soft errors are generated by cosmic particles, energetic neutrons, and alpha particles hitting the surface of silicon devices.

Device scaling significantly affects the susceptibility of integrated circuits to soft errors [34]. As the feature size shrinks, the amount of charge per device decreases thereby enabling a particle strike to be much more likely to cause an error. As a result, particles of lower energy, which are far more plentiful, can generate sufficient charge to cause a soft error. Hence, in the absence of error correction schemes, the system error rate will grow in direct proportion to the number of bits on the chip. Thus, while Moore's Law gives an exponential increase in the transistor count, this growth comes at the cost of an exponential increase in the error rates for unprotected chips [6], [27].

Field Programmable Gate Arrays (FPGAs) are utilized in many applications such as networking, information systems, and embedded applications due to their high performance, low Non Recurring Engineering (NRE) cost, and fast time to market. In SRAM-based FPGAs, the configuration of the design mapped into the FPGA chip is stored in SRAM cells, called configuration bits. FPGAs are far more vulnerable to SEUs compared to Application-Specific Integrated Circuit (ASIC) implementations [13]. This is mainly due to the fact that the majority of FPGA chip real estate is dedicated to memory cells that maintain the state of the configuration and the circuit, as well as block memory arrays. Memory elements are by far the most vulnerable components to soft errors (compared to combinational logic) [24].

Memory elements in an FPGA device can be divided into two categories: 1) *configuration* and 2) *user* bits. Configuration bits are used to specify the particular circuit mapped into the FPGA, whereas the user bits, such as flip-flops (FFs) or on-chip memory arrays, hold the current state of the circuit. After loading a design into an FPGA, the contents of the configuration bits are supposed to remain unchanged, while the contents of user bits can be changed on any clock cycle. The majority (more than 99%) of memory bits in an FPGA are configuration bits and therefore, the probability of soft errors in configuration bits is much greater than that in user bits. Moreover, a particle hit on a configuration bit causes a permanent error.

One of the key design points in high-end information systems is availability. From the users' stand point, they want to be sure that data is never lost (*reliability*). Also, the application which heavily depends upon that data is required to remain available to access (*availability*). Therefore, a considerable amount of redundancy, error checking (parity), and error correction has been integrated in the design of these systems [15]. Using hardware and/or software redundancy techniques, soft errors are managed by the system to prevent any corruption of the data.

FPGAs are commonly used in the design of state-of-the-art information systems. FPGAs are also frequently used in the implementation of the adapters that interface to either hosts or disk arrays. Hence, the soft error reliability of these FPGAs is critical in the overall dependability of information systems.

To achieve a reasonable balance between reliability and performance, the effect of soft errors at the system level and the contribution of each component to the overall soft error rate

Manuscript received March 15, 2007; revised June 13, 2007.

H. Asadi and K. Granlund are with the Reliabilitiy Engineering Department, EMC Corporation, Hopkinton, MA 01748 USA (e-mail: Asadi_Hossein@emc.com).

M. B. Tahoori and D. Kaeli are with the Electrical and Computer Engineering Department, Northeastern University, Boston, MA 02115 USA.

B. Mullins is with the RSA Division, EMC Corporation, Hopkinton, MA 01748 USA.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNS.2007.910426

0018-9499/$25.00 © 2007 IEEE

Authorized licensed use limited to: Northeastern University. Downloaded on April 21, 2009 at 09:19 from IEEE Xplore. Restrictions apply.

of the system needs to be precisely analyzed. Tools are needed to be able to identify the most vulnerable components in the system. Using such analysis and given a redundancy budget, the most vulnerable components can be protected in the most effective manner (using hardware or software redundancy).

In this paper, we focus on soft error rate estimation of FPGAs used in information systems designs. We present an accurate analytical *Soft Error Rate* (SER) estimation methodology for designs mapped into SRAM-based FPGAs. Unlike previous work which are based on fault injection and random vector simulations [2], [8], [11], [13], [23], [32], the presented approach is analytical and does not rely upon vector simulations. There have been some analytical methods [33], [35] to estimate the dependability of designs implemented on SRAM-based FPGAs. These methods do not take into account the exact *error propagation probability* (EPP) of the SEUs from the error sites to system outputs. Using our approach, we first compute the netlist error rate based on the particular FPGA resources used for the given mapped design (obtained from detailed placement and routing information). Then, error propagation probabilities are computed using a gate-level netlist. Based on the netlist error rate and the error propagation probabilities, the failure rate of a particular mapped design is estimated. The proposed SER methodology is independent of the underlying FPGA physical architecture and can be easily applied to various FPGA architectures.

This work complements our previous work [5] by modeling the effect of both short and open errors in the routing configuration bits. In our previous study, only the effect of open errors in routing configuration bits was considered. In this paper, we have also extended our previous analytical SER estimation framework presented in [5] by computing SER of configuration bits in multiple clock cycles after a particle strike. In our previous work, error propagation in configuration bits were computed from the error site to flip-flops and primary outputs. However, in general, not every erroneous flip-flop causes a system failure since the error may be masked and not propagated to a primary output. In this work, we extend our formulation to compute exact SER of configuration bits from the error site to primary outputs in multiple clock cycles after a particle strike.

We have also validated our methodology with field data results. We present failure rate data as obtained from live information systems in the field for a particular FPGA-based controller. We need to stress that this is not experimental data in the sense that we are not creating any artificial environment for our validation study. The data are raw error logs taken. We compare two different SEU rates: one rate that is derived from a comprehensive field failure data analysis and another obtained from our analytical tool. To our best knowledge, this is the first attempt to validate a system-level soft error modeling tool using real and comprehensive field failure data. Preliminary results of this case study were presented in [28].

The rest of the paper is organized as follows. Section II explains the previous SER estimation techniques. Section III describes the error models of SRAM-based FPGAs. In Section IV, the failure rate estimation technique for FPGA designs is presented. In Section V, a case study that applies SER estimation to an FPGA-based controller resident in a information system is described. A validation study is also presented, comparing field failure data with results from our analytical tool. Finally, Section VI concludes the paper.

II. Previous Work

EPP is one of the main factors for SER estimation of the circuit nodes [25]. Previous work on soft error rate estimation is mainly simulation-based, radiation-based, or a combination of both [1], [2], [7]–[9], [11]–[13], [19], [23], [32]. Most of these methods have been based on *Fault Injection* (FI) strategies. To compute the EPP of a node 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. The process of fault injection in these methods is done either using simulation or radiation equipment. These steps make FI approaches both very time-consuming and inaccurate. Furthermore, these approaches cannot be used during design phases since they need physical implementation.

There is a plethora of radiation-based methods which use accelerated radiation testing to measure the sensitivity of FPGA devices to SEUs [7]–[9], [11], [19], [36]. In these approaches, a prototype of the system under study is exposed to a flux of radiation, originated either by radioactive sources or by particle accelerators. These particles interacts with both the configuration memory and the design memory elements. The FPGA under test is continuously stimulated by a given set of input stimuli and the system outputs are compared with expected output values. As a major drawback, the radiation-based methods are very expensive and they are not commonly used. These methods are mainly used for device characterization, not for SER estimation of a particular mapped design. Additionally, radiation-based methods cannot be used during early design phases. Lastly, radiation is capable of permanently damaging the FPGA under study, making a hard error in the device which can invalidate results.

The methods presented in [1], [2], [12], [13], [23], and [32] compute the SER of an implemented design based on the alteration of the configuration bitstream. The device is configured for every faulty bitstream, i.e., one configuration bit is flipped for each workload. Then, the system is run several clock cycles with input stimuli to compare the results with the golden-run results. These methods can be further classified into two groups: the first group [1], [2], [32] gathers and compares the results in a host system; the main drawback is that it takes too much time to do experiments for all possible faults. The second group [12], [13], [23] uses one FPGA for the faulty run, one FPGA for the golden-run, and another FPGA for the comparison of results. In order to implement and evaluate larger designs, a new prototype board equipped with higher density FPGAs is required. Finally, as FPGA-based designs have a larger number of candidate fault locations compared to similar ASIC designs with the same density, FI techniques are more time-consuming for FPGA designs than for ASIC designs.

A simulation-based analysis of SEU effects in SRAM-based FPGAs is presented in [38]. In this approach, the physical layout sensitivity to SEUs and the application mapped into the FPGA are analyzed independently. While physical layout sensitivity

Authorized licensed use limited to: Northeastern University. Downloaded on April 21, 2009 at 09:19 from IEEE Xplore. Restrictions apply.

to SEUs is examined using beam-testing, the mapped application is analyzed using FI techniques. The predicted results using this approach were within a factor of two of the measured results [38].

Ceschia *et al.* [10] have classified the sensitivity of configuration memory to heavy-ion-induced SEUs using irradiation testing. They have classified the effects of SEUs on FPGA resources into four classes, 1) PIP-open, 2) PIP-short, 3) PIP-antenna, and 4) Other. They have concluded that 61.5% of SEUs affect the "other" category. Throughout the experiments, they utilized a Xilinx Virtex XCV300 FPGA and have used various ion species from carbon to iodine featuring different energy levels.

Graham *et al.* [13] have examined the failure modes of FPGA-based designs. They have shown that SEUs in SRAM cells can result in five main categories of design modifications, errors in mux select lines, programmable interconnect point states, buffer enables, *Look-Up Table* (LUT) cells, and control bits. Their analysis found that failures due to routing structures account for 78%-84.8% of the failures; the remaining percentage of failures (15.2%-22%) were due to upsets in control bits and LUT value changes.

Wirthlin *et al.* [16], [26], [39] have developed a fault injection tool that identifies the sensitive configuration bits of the device for any given FPGA design. This tool operates by artificially injecting faults within the configuration bitstream and monitoring the behavior of the device. By comparing the behavior of the device under test against a golden device, they can determine when the device behavior changes. By testing every configuration bit of the device, a detailed sensitivity profile of a given design can be created. This fault injection tool has been used to characterize the sensitivity of many FPGA designs [16], [26].

Wirthlin *et al.* [16], [26], [39] have also introduced a new way to categorize the sensitive configuration bits by separating them into two categories: persistent and non-persistent. A non-persistent configuration bit is a sensitive configuration bit that will cause a design fault when upset through radiation. This fault will introduce functional errors. When the non-persistent configuration bit is repaired through configuration scrubbing, however, the design returns to normal operation and all previously induced functional errors will disappear. A persistent configuration bit is a sensitive configuration bit that will cause a design fault when upset. However, even after repairing persistent configuration bits through configuration scrubbing, the FPGA circuit does not return to normal operation. Upsets of persistent configuration bits introduce functional errors which persist after the bit is repaired through scrubbing. Upsets of persistent bits put the design into an incorrect state that cannot self-correct. In this case, a global reset is needed to return the circuit to a proper state, or normal operation.

Sterpone *et al.* [33], [35] have proposed an analytical approach to estimate the dependability of *Triple Modular Redundancy* (TMR) designs implemented on SRAM-based FPGAs. In their proposed approach, the routing resources are investigated in order to eliminate the effect of single-point of failure in the TMR implementation. The routing structure of the FPGA is described by a colored graph where vertices model logic blocks and switch boxes while the edges correspond to wiring segments or PIPs. The vertices corresponding to the FPGA resources that are used to implement the circuit are colored. In the case of TMR, different colors are used to mark the vertices of each circuit replica, as well as the majority voter. In the proposed method, they also utilize the device physical layout information and extract the number of potential SEUs that may occur in the circuit. The presented approach, however, does not take into account the exact error propagation probability of the SEUs from the error sites to system outputs.

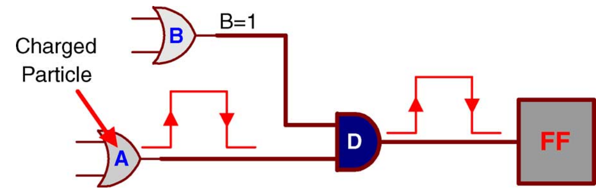

Fig. 1. A charged particle affects one of inputs of the AND gate and makes a bit-flip error.

III. FPGA Error Models

There are two major types of memory resources in FPGAs, 1) user bits, and 2) configuration bits. An SEU in a user bit causes a transient error, whereas an SEU in a configuration bit leads to a permanent error.

A. Transient Errors

Transient errors do not alter SRAM configuration bits but they do affect user-defined logic, flip-flops, and the combinational logic of the *Configurable Logic Blocks* (CLBs). A charged particle can hit the combinational logic within the CLBs and create a transient voltage pulse [called, Single Event Transient (SET)]. This SET can be propagated to sequential logic and generate a bit-flip error. Fig. 1 illustrates how an SET introduces a bit-flip error in a flip-flop. It has been shown that in ASIC designs, combinational logic is less susceptible to soft errors than memory elements [22], [34]. This is because the combinational logic provides some natural resistance to soft errors, including logical masking, electrical masking, and latch-window masking [34].

A charged particle may also directly affect the contents of a flip-flop and user-defined memory elements and cause a single event upset. The content of the flip-flop will remain erroneous until it is rewritten with new data or it is corrected by some appropriate error detection and correction logic.

B. Permanent Errors

An SEU that changes a configuration SRAM cell has a permanent effect until the original configuration bitstream is redownloaded into the FPGA. This type of error is the major error type found in FPGAs because the number of SRAM cells dominates all of the user-defined memory elements. Typically, the number of SRAM configuration cells are more than 98% of all memory elements inside an FPGA [40], [41].

The configuration memory bits are categorized into *sensitive* and *non-sensitive* bits according to their vulnerability to SEUs. An SEU in a sensitive configuration bit affects the functionality of the particular circuit mapped into the FPGA; non-sensi-

Authorized licensed use limited to: Northeastern University. Downloaded on April 21, 2009 at 09:19 from IEEE Xplore. Restrictions apply.

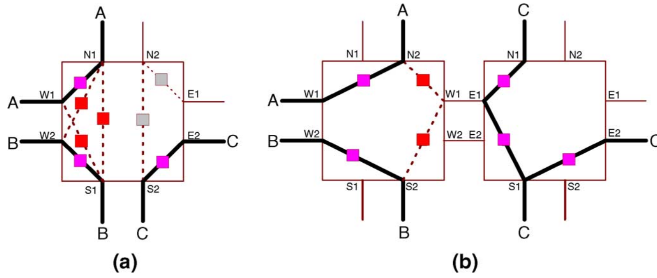

Fig. 2. The impact of SEUs on routing signals.

tive bits act as "don't care" configuration bits for that particular mapped design. Hence, the sensitivity of each configuration bit is *application-dependent*.

Permanent errors are classified into routing errors, LUT bit-flips, and control/clocking bit-flips. We briefly review these errors here. These errors are well described in [5]. Programmable Interconnect Points (PIPs), multiplexers, and buffers constitute the programmable routing network of a segmented-routing FPGA (e.g., Xilinx Virtex FPGAs). Routing resources can be *inter-CLBs* or *intra-CLB*. An inter-CLB routing signal connects two or more CLBs. Those that are 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-CLB routing. When an SEU changes the configuration routing bit it causes an open switch, a shorted switch, or a bridging error (wired-or, wired-and), as shown in Fig. 2.

Another type of permanent error is a bit-flip within an LUT configuration bit. A look-up table implements a logic function by storing all values in the truth table. A bit-flip within an LUT can change the LUT original functionality.

There are also some control bits inside CLBs and input-output blocks (IOBs) that are used to determine miscellaneous functionality. One example is the control bits that determine whether the LUT functions as a look up table, a dual-ported RAM, or a programmable shift register. A bit-flip on the control/clocking bits can drastically affect the functionality of the mapped design.

IV. Failure Rate Estimation of FPGA Designs

There are some differences between computing the failure probability for an ASIC design and an FPGA-based design:

- In ASIC designs, only the propagation of an erroneous value from the error site to primary outputs (POs) or flip-flops (FFs) needs to be considered. However, in FPGA designs, the activation probability as well as the propagation probability are required for failure rate estimation. This is because in FPGAs a failure only occurs if an erroneous node (e.g., an LUT cell) is first activated by inputs and then the error is propagated to the POs or FFs (permanent errors).
- In FPGAs, the errors occurring in the configuration bits remain unchanged during the next clock cycle after the bit-flip. Thus, the same failure probability is valid for two clock cycles. In ASIC designs, however, if an erroneous value due to an SEU is masked and the error is not propagated to the outputs (POs or FFs), the effect of that SEU will be completely transparent to the system.
- The error sites in ASICs are mainly logic gates rather than routing signals. In FPGAs, routing signals (controlled by SRAM cells) constitute more than 70% of the total sensitive SRAM bits. [13].
- In FPGAs, if an SEU flips the contents of a configuration bit, an erroneous value can be propagated from the error site to the system outputs without any attenuation (no electrical masking). However, electrical masking is one of the key factors that causes the combinational logic in ASIC designs to be less susceptible to soft errors (compared to memory elements) [34]. 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 a point where the SEU does not affect system outputs.

To compute the failure rate of a design mapped into an FPGA, we follow these steps. First, we compute the netlist failure probability. In this step, the gate-level netlist of the mapped design is used. Second, we compute the error rate of all nodes of the circuit. This step is performed based on the detailed FPGA placement and routing information of the mapped design (i.e., the detailed information regarding the used and unused FPGA resources). Finally, the system failure rate is computed based on these two steps. The details of these steps are subsequently provided. Note that in this paper, we focus on the FPGA SER due to errors in configuration bits. As mentioned in Section I, the majority (more than 99%) of memory bits in an SRAM-based FPGA are configuration bits and therefore, the probability of soft errors in configuration bits is much greater than that in user bits. A methodology to compute FPGA SER due to user-bits has been presented in [5].

A. Error Propagation Probabilities (PP_i)

While particle flux is uniformly spread across the entire system, the probability of an erroneous value being observed at the system outputs depends heavily on which node the particle strikes and the values of other nodes in the circuit at that time (i.e., the system state).

In our proposed approach for error propagation probability computation, we use the signal probabilities of all nodes in the combinational part and the topological structure of the circuit [3], [4]. The signal probability (SP) of a signal line is the probability of a logic value 1 (versus 0) on that line [31]. In our approach, the structural paths from each error site to all reachable outputs and flip-flops are extracted. These paths are then traversed to trace and compute the error propagation probabilities to reachable outputs or flip-flops. Using this approach, the following probabilities are computed:

- PP_{Gi}: Error propagation probability from error site G_i to any primary output.
- PP_{Fi}: Error propagation probability from flip-flop FF_i to any primary output.
- $PP_{Gi \to Oj}$: Error propagation probability from error site G_i to primary output O_j.
- $PP_{Gi \to FFj}$: Error propagation probability from error site G_i to flip-flop FF_j.
- $PP_{FFi \to FFj}$: Error propagation probability from flip-flop FF_i to flip-flop FF_j.
- $PP_{FFi \to Oj}$: Error propagation probability from flip-flop FF_i to primary output O_j.

We use these probabilities in the following subsections. More details of this approach can be found in [3], [4].

Authorized licensed use limited to: Northeastern University. Downloaded on April 21, 2009 at 09:19 from IEEE Xplore. Restrictions apply.

B. Netlist Impact Probability (NIP)

The netlist impact probability (*NIP*) is the probability that the error site is activated by the inputs and then propagated to the outputs. *NIP* depends on the error model and the circuit topology. In general, NIP_i is the product of the activation probability of node i (AP_i) and the propagation probability of that node (PP_i). We use signal probabilities for computing AP_i. Computation of PP_i has been presented in Section IV-A.

In the case of open and stuck-at errors, NIP_i can be computed according to (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 (signal probability, SP_i, is used for the activation probability) and then propagates it to the outputs (PP_i). Note that $PP_i(0) = PP_i(1)$.

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

PP_i : *Propagation probability*

SP_i : *Signal probability*

For wired-AND bridging errors (between nets i and j), NIP_i can be computed according to (2). The first term of the equation expresses the probability of node i being 1 and node j being 0. The second term calculates the probability of node i being 0 and node j being 1.

$$NIP_i = [SP_i \times (1 - SP_j) \times PP_i(0)] + [(1 - SP_i) \times SP_j \times PP_j(0)]. \quad (2)$$

We use the same approach to compute NIP_i for wired-OR bridging errors between nets i and j, as shown in (3).

$$NIP_i = [SP_i \times (1 - SP_j) \times PP_j(1)] + [(1 - SP_i) \times SP_j \times PP_i(1)]. \quad (3)$$

For a bit-flip in one of the LUT cells (cell i), NIP_i is computed according to (4). To compute the failure probability due to the LUT SRAMs, we use the propagation probability of the LUT output. NIP_i is computed as the product of the propagation probability of the LUT output and the activation probability of the SRAM cell from LUT inputs (which is computed during SP estimation).

$$NIP_i = AP_i \times PP_i(LUT_{out}). \quad (4)$$

Therefore, based on PP_{Gi}, PP_{Fi}, $PP_{Gi\to Oj}$, $PP_{Gi\to FFj}$, $PP_{FFi\to FFj}$, and $PP_{FFi\to Oj}$, we will compute NIP_{Gi}, NIP_{Fi}, $NIP_{Gi\to Oj}$, $NIP_{Gi\to FFj}$, $NIP_{FFi\to FFj}$, and $NIP_{FFi\to Oj}$, respectively.

C. Node Error Rate (NER_i)

PP_i and NIP_i depend only upon the gate-level netlist of the mapped design. In contrast, the node error rate depends on the detailed FPGA placement and routing information of the design. For each gate-level circuit node i, NER_i is defined as the permanent-error rate of node i. NER_i is calculated based on the raw error rate of the device, the error model, and the number of SRAM configuration bits used for implementing node i in the FPGA. The error rate of a node is directly proportional to the number of SRAM configuration cells controlling that node. Thus, nodes with more configuration bits have higher error rates.

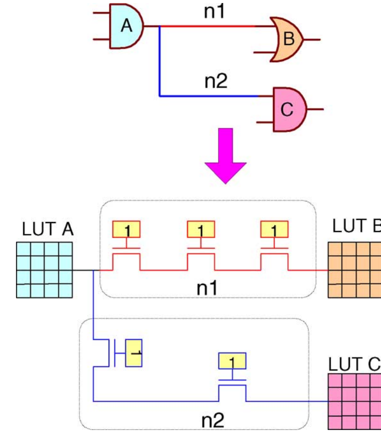

Fig. 3. Nodes n_1 and n_2 with different error rates.

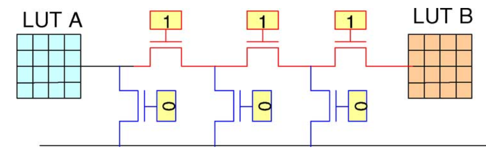

Fig. 4. Nets implemented with different numbers of switches.

For example, consider two nodes, n_1 and n_2, which consist of 3 and 2 PIPs, respectively (Fig. 3). 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. NER_i is computed as shown in (5). In this equation, *nsrams* is the total number of possible susceptible SRAM cells which can occur on node i. For example, *nsrams* would be equal to the value 1 for each LUT cell since it consists of only one SRAM; for multiplexers, *nsrams* would be equal to the number of select bits. Lastly, for a routing node, *nsrams* would be directly proportional to the number of ON/OFF switches connected to that node. To illustrate this, the permanent-error rate of node *AB*, shown in Fig. 4, is equal to six times the raw error rate of an SRAM-cell. In our method, we compute NER_i and NIP_i for both short and open errors. In this example, $NER_i(short) = 3 \times fpb$ and $NER_i(open) = 3 \times fpb$, where *fpb* (FIT rate per Bit) is the raw FIT rate of an SRAM cell. Accordingly, we compute $NIP_i(short)$ and $NIP_i(open)$ separately.

$$NER_i = fpb \times nsrams. \quad (5)$$

fpb : *Raw FIT rate of an SRAM cell*

nsrams : *Number of possible susceptible SRAMs*

Authorized licensed use limited to: Northeastern University. Downloaded on April 21, 2009 at 09:19 from IEEE Xplore. Restrictions apply.

The raw error rate of an SRAM cell (*fpb*) depends on the device characteristics and the flux encountered by the device. The raw error rate can be expressed in terms of either MTBF (Mean Time Between Failures) or FIT (Failure in Time). FIT is inversely proportional to MTBF and is equal to one failure in a billion hours (10^9). Designers usually work with FIT units because they are additive, unlike MTBF. The FIT rates for FPGA SRAMs differ from commercial SRAMs [20]. Current predictions show that typical FIT rates for latches and commercial SRAM cells (measured at sea level) vary between 0.001–0.01 FIT/bit [14], [18], [30]. The FIT rates for the device considered in this study have been reported to be from 50 FIT/Mbit (0.00005 FIT/bit) up to 1000 FIT/Mbit (0.001 FIT/bit) for 0.15 μm, 0.13 μm, and 90 nm technologies [20], [21]. The FIT/bit rate increases with elevation. At 10 km, the FIT/bit is approximately 100 × higher than at sea-level [42].

D. Failure Rate Computation

After the computation of NER_i and NIP_i, the system failure probability due to node i can be computed as follows:

$$System\ failure\ rate\ SFR_i = NER_i \times NIP_i. \quad (6)$$

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

$$SFR_{chip} = \sum_{i=1}^{n} SFR_i = \sum_{i=1}^{n} NER_i \times NIP_i. \quad (7)$$

The above expressions compute the system failure probability only for the first clock cycle after the particle hit. In general, the particle can impact system correctness during any c clock cycles after a particle hit.

Here, we present an analytical framework to estimate SERs in multiple clock cycles after a particle strike. In general, not every erroneous flip-flop causes a system failure since this error may be masked or may not be propagated to a primary output. In other words, system failures occur only when an erroneous value is propagated to a primary output.

An erroneous value in an internal node or a flip-flop can be propagated to a primary output either directly or indirectly through other flip-flops. To formalize this analysis, we define *SGF*, *SGO*, *SFO*, and *SFF* as follows. *SGF* is an $m \times n$ matrix, where SGF_{ij} is the probability of an error in flip-flop FF_j given an internal node (gate) G_i is erroneous. In other words, SGF_{ij} equals to $NIP_{Gi \rightarrow FFj}$, which is the netlist impact probability from G_i to FF_j. m is the number of internal nodes (gates) and n is the number of flip-flops. *SGO* is an m–element column vector, where SGO_i is the probability of an error in the primary outputs given gate G_i is erroneous. SGO_i equals to NIP_{Gi}. *SFO* is an n–element column vector, where SFO_i is the probability of an error in the primary outputs given that flip-flop FF_i is erroneous. SFO_i equals to NIP_{FFi}. And finally, *SFF* is an $n \times n$ matrix, where SFF_{ij} is the probability of an error in flip-flop FF_j given flip-flop FF_i is erroneous. SFF_{ij} equals to $NIP_{FFi \rightarrow FFj}$, which is the netlist impact probability from FF_i to FF_j.

$$\mathbf{SFF} = \begin{pmatrix} NIP_{FF1\rightarrow FF1} & NIP_{FF1\rightarrow FF2} & \cdots & NIP_{FF1\rightarrow FFn} \\ NIP_{FF2\rightarrow FF1} & NIP_{FF2\rightarrow FF2} & \cdots & NIP_{FF2\rightarrow FFn} \\ \vdots & \vdots & \ddots & \vdots \\ NIP_{FFn\rightarrow FF1} & NIP_{FFn\rightarrow FF2} & \cdots & NIP_{FFn\rightarrow FFn} \end{pmatrix}$$

$$\mathbf{SGF} = \begin{pmatrix} NIP_{G1\rightarrow FF1} & NIP_{G1\rightarrow FF2} & \cdots & NIP_{G1\rightarrow FFn} \\ NIP_{G2\rightarrow FF1} & NIP_{G2\rightarrow FF2} & \cdots & NIP_{G2\rightarrow FFn} \\ \vdots & \vdots & \ddots & \vdots \\ NIP_{Gm\rightarrow FF1} & NIP_{Gm\rightarrow FF2} & \cdots & NIP_{Gm\rightarrow FFn} \end{pmatrix}$$

$$\mathbf{SFO} = \begin{pmatrix} NIP_{F1} \\ NIP_{F2} \\ \vdots \\ NIP_{Fn} \end{pmatrix}$$

$$\mathbf{SGO} = \begin{pmatrix} NIP_{G1} \\ NIP_{G2} \\ \vdots \\ NIP_{Gm} \end{pmatrix}.$$

Theorem 1: The probability of a system failure c clock cycles after the SEU event in G_i (i.e., given G_i is erroneous), $P^c(SF|G_i)$, is calculated as follows:

$$\begin{aligned} &P(SF\ at\ cycle\ c|G_i\ erroneous): \\ &P^c(SF|G_i) = SGO(i), \quad if\ \ c=1 \\ &P^c(SF|G_i) = P^{c-1}(SF|G_i) \\ &+i^{th}\ element\ of(SGF \times SFF^{c-2} \times SFO), \ \ if\ c>1. \end{aligned} \quad (8)$$

Proof: Assume that the contents of G_i is erroneous. The probability of a system failure during the first clock cycle equals to $P^1(SF|G_i) = SGO(i)$, based on the definition of $P^c(SF|G_i)$ (the basis of induction).

An error can be propagated to an output in two clock cycles after the SEU event if the error is propagated from G_i to flip-flop FF_j in the first clock cycle, and then propagated from FF_j to an output in the second clock cycle. The error can be also directly propagated from G_i to an output during the second cycle. Therefore, the failure probability during the second clock cycle is equal to $SGO(i) + i^{th}\ element\ of\ (SGF \times SFO)$. If we rewrite (8) for $c = 2$, we will have:

$$\begin{aligned} &P^c(SF|G_i)(c=2) \\ &= P^1(SF|G_i) + i^{th}\ element\ of\ (SGF \times SFF^0 \times SFO) \\ &= SGO(i) + i^{th}\ element\ of\ (SGF \times SFO). \end{aligned}$$

Therefore, Theorem 1 holds for $c = 2$.

Now, we consider the failure probability during the third cycle. Here, there are three different scenarios that can happen: 1) An SEU can be propagated from the error site (G_i) to an output during the third cycle with probability of $SGO(i)$, 2) An SEU can be propagated from G_i to flip-flop FF_j in the second

Authorized licensed use limited to: Northeastern University. Downloaded on April 21, 2009 at 09:19 from IEEE Xplore. Restrictions apply.

clock cycle, and then propagated from FF_j to an output in the third clock cycle with the probability of $SGF(i) \times SFO(j)$, 3) Finally, an SEU can be propagated from G_i to flip-flop FF_j in the first clock cycle, then propagated from flip-flop FF_j to flip-flop FF_k at the second cycle, and lastly propagated from FF_k to an output in the third clock cycle with the probability of $i^{th}\ element\ of\ (SGF \times SFF \times SFO)$. Thus, the failure probability during the third clock cycle equals to

$$SGO(i) + i^{th}\ element\ of\ (SGF \times SFO) + i^{th}\ element\ of\ (SGF \times SFF \times SFO).$$

If we rewrite (8) for $c = 3$, we will have:

$$\begin{aligned} &P^c(SF|G_i)(c=3) \\ &= P^2(SF|G_i) + i^{th}\ element\ of\ (SGF \times SFF^1 \times SFO) \\ &= SGO(i) + i^{th}\ element\ of\ (SGF \times SFO) \\ &\quad + i^{th}\ element\ of\ (SGF \times SFF \times SFO). \end{aligned}$$

Therefore, Theorem 1 holds for $c = 3$.

In the general case (clock cycle c), the $P^c(SF|G_i)$ can be written down as follows:

$$\begin{aligned} P^c(SF|G_i) =& SGO(i) \\ &+ i^{th}\ element\ of\ (SGF \times SFF^0 \times SFO) \\ &+ i^{th}\ element\ of\ (SGF \times SFF^1 \times SFO) \\ &+ i^{th}\ element\ of\ (SGF \times SFF^2 \times SFO) \\ &+ \ldots \\ &+ i^{th}\ element\ of\ (SGF \times SFF^{c-3} \times SFO) \\ &+ i^{th}\ element\ of\ (SGF \times SFF^{c-2} \times SFO) \\ \Longrightarrow P^c(SF|G_i) =& P^{c-1}(SF|G_i) \\ &+ i^{th}\ element\ of\ (SGF \times SFF^{c-2} \times SFO). \end{aligned}$$

After computing system failure probabilities for any particular clock cycle c after the SEU event, the failure probability for this period (i.e., from the clock cycle at which the bit-flip occurs to c clock cycles after that) is computed as follows:

$$\begin{aligned} &P_1^c(SF|G_i) \\ &= P^1(SF|G_i) \\ &\quad + (1 - P^1(SF|G_i)) \times P^2(SF|G_i) \\ &\quad + (1 - P^1(SF|G_i)) \times (1 - P^2(SF|G_i)) \times P^3(SF|G_i) \\ &\quad + \cdots \\ &= \sum_{k=1}^{c} \left(P^k(SF|G_i) \prod_{l=1}^{k-1} (1 - P^l(SF|G_i)) \right). \end{aligned} \quad (9)$$

After computing $P_1^c(SF|G_i)$ for all nodes, the system failure rate can be computed as follows:

$$SFR_{chip} = \sum_{i=1}^{n} NER_i \times P_1^c(SF|G_i). \quad (10)$$

Note that the complexity of the simulation-based FI method increases exponentially with c, making simulation-based analysis intractable for large sequential circuits. However, our approach requires only a matrix multiplication to compute the system failure rate in the subsequent clock cycles, and hence, its time complexity is linear with c.

Matrices *SFF* and *SGF* and vectors *SFO* and *SGO* are computed using the propagation probability computation approach presented in Section IV-A. Each column of *SFF* and *SGF* (as well as the corresponding entry in *SFO* and *SGO*) is computed by a traversal of the circuit from the corresponding flip-flop or gate to system outputs and flip-flops.

E. Accuracy of the Proposed Theorem

In the above formulation, it has been assumed that the entries in matrices *SFF* and *SGF* are independent of each other. Based on this assumption, we have used additive probabilities in the above equations. This may introduce some inaccuracy to the proposed theorem. As an example, assume that G_i is erroneous and also assume that $NIP_{G_i \to FF_1} \times NIP_{FF_1} = 0.15$ and $NIP_{G_i \to FF_2} \times NIP_{FF_2} = 0.05$. The probability of system failure in the second cycle due to error propagation through FF_1 and FF_2 is calculated as $0.15 + 0.05 = 0.20$. The exact probability can be any probability between 0.15 and 0.20, and depends upon the structural dependency of FF_1 and FF_2 on G_i. Therefore, the calculated probabilities computed by (8) are not exact. However, the experimental results (Section V-E) show this theorem provides very close approximations (there is an 81% overlap in FIT rate range obtained with our analytical modeling framework and the field failure data studied). This is due to several reasons. First, a majority of the entries in matrices *SFF* and *SGF* are either zero or a very close number to zero (less than 0.05). In particular, since each node has very few reachable flip-flops (i.e., an error can only propagate to a very limited number of flip-flops), most entries of matrices *SFF* and *SGF* are 0. This means that most structural paths are independent. Second, since there can be several logic stages between flip-flops, this reduces the dependency between entries in matrices *SFF* and *SGF*.

Note that since the *SSF* entries are very small, matrix multiplication of *SFF* causes that the entries of SFF^i to be increasingly smaller than the entries of SFF^{i-1}. Therefore, it is expected that the term i^{th} element of $(SGF \times SFFc \times SFO)$ to become (or tend to) zero for very large c. This means that (9) saturates for large c. In our case study, we have noted that this equation saturates for c values of 60 to 100 clock cycles. In applications where configuration bits are periodically checked and corrected (called, *error checking and recovery*), the c factor depends on the error checking period. Frequent error checking and recovery will reduce the FPGA SER. The soft error susceptibility analysis for error checking and recovery applications has been described in [5].

V. Case Study

A. Embedded Control Unit in Information Systems

In this study we consider FPGAs that are commonly used in the design of high performance information systems. These sys-

Authorized licensed use limited to: Northeastern University. Downloaded on April 21, 2009 at 09:19 from IEEE Xplore. Restrictions apply.

tems typically hold several hundred disks, which can be protected via *Redundant Array of Inexpensive Disks* (RAID) protocols (e.g., RAID-1, RAID-5). The internal architecture provides for a high degree of redundancy so that a failure in any bus component does not disconnect any component from the system. One of the components that connects to the buses is the *Logical Unit Module* (LUM). Multiple LUMs connect the server host to the internal buses of the disk arrays. These LUMs are used to manage the memory caches.

Each LUM has several *Embedded Control Units* (ECUs); each ECU controls multiple microprocessors, DRAMs, and L2 caches. All ECUs have been implemented on an SRAM-based FPGA, and each ECU acts independently. This FPGA component is the target of our validation study.

The ECU design has been implemented using a very popular and very large commercial FPGA device. Table IV depicts the FPGA utilization of this device for the ECU design. As seen in this table, the ECU design uses 99.9% of the available slices and 73.4% of the available *Look-up Tables* (LUTs).

B. Field Data Analysis

Field data analysis is traditionally used in industry to evaluate system failure rates. In our case study, this mechanism has been utilized to calculate the FIT rate for the Logical Unit Module (LUM) based solely upon SEUs observed in the field. Note that this study is based on actual field failure data from the information system vendor, using the available repository of field data as collected by the manufacturer. The system provided automated reporting of errors, and field engineers also went out into the field to verify these errors.

Specifically, SEUs that occurred on distinct FPGA components (we will refer to these as FSEUs) contained within the specified LUM were investigated. It should be noted that this field analysis was not limited in scope to one particular geographic location; data was collected from *all* functioning systems distributed in more than 35 countries, spanning regions from all across the globe. Thus, the analyzed field data represents a *global* distribution. In total, twenty-eight months of field data was analyzed, including more than 750,000 FPGAs. The information needed for this analysis was available using the error reporting systems implemented in the field.

Initially, all errors in the LUM that were flagged in the field as *No Evidence of Failure* (NEOF) were collected and analyzed. NEOFs can describe a range of errors including:

- state-dependent logic or timing errors;
- software-based errors;
- signal cross-talk;
- particle-induced soft errors (SEUs).

In total, $\sim$ 7900 NEOF events were analyzed.

The next step in this process was to analyze the error code associated with each NEOF event. The information system manufacturer identified six specific error codes that have been verified in the field to be associated with FSEUs in the investigated FPGA devices. The way that these error codes were identified is as follows: once a trend of errors has been seen in the field, the information system manufacturer actually goes out into the field and scans out the FPGA to compare the bit pattern to the original configuration pattern. When a bit difference is found, the conclusion is that this bit flip was definitively caused by an FSEU. Note that even though an error event may possess one of the six error codes, it still may not have been caused by an FSEU. We will refer to errors possessing these six error codes as *Probable FSEUs*.

An additional three error codes were also identified as being potentially associated with FSEUs in the LUM, though field studies have not confirmed this yet (we will refer to error events possessing these three error codes as *Potential FSEUs*). We will label any NEOF event as a *Possible FSEU* if it possesses one of these nine error codes.

Furthermore, when an FSEU is confirmed to have occurred, there is a unique relationship between the observed error and the affected logic within the FPGA design. Particularly, of the 7900 NEOFs found in the field, further analysis is performed by inspecting the associated error log and by also reading out the faulty FPGA device. This allows us to tie specific error codes found in the logs to specific bit flips in the programmable bits of the FPGA. This also allowed us to analyze the partial contributions to the overall SEU rate of the various sub-functions implemented in the FPGA, and then correlate these to the contributions of individual data-paths in our modeling environment. This is defined as the FSEU's failure signature. Over time, these signatures are documented, detailing both the specific field errors that represent possible FSEUs and their distinct failure signatures.

In general, FSEUs in the LUM have been observed to manifest themselves in several ways such as CRC errors, parity errors, timeout errors, and data miscompares. A specific example of an FSEU is a parity bit being set during an interrupt operation (e.g., a situation that can only arise by the internal logic of the FPGA being incorrect).

After all LUMs that were categorized as NEOF were collected and analyzed, all potential FSEUs were identified according to their specific field error code as described above. This reduced the analysis space by an order of magnitude.

After completing this step, the focus of our field study was then reduced in scope to two specific FPGAs within the LUM. This decision was motivated by a high degree of understanding of the effects of FSEUs on these components and the significant role that these FPGAs play within the LUM.

The final step in identifying all FSEUs that occurred within the two FPGAs of interest was to inspect the log file provided with each *Possible FSEU*. Each error's log file contained all of the system information at the time when the error occurred, in addition to information that described how the system behaved both before, and after, the error was observed. In order for an error to be classified as a FSEU, it was required that the error's log file contain the failure signature that specifically pointed to a bit flip in the configuration bits in the FPGA (bit flips can also occur in the non-configuration bits of the FPGA, but in these designs the physical space in both FPGAs is heavily dominated by configuration bits [5]). There are also cases where the signature confirms that the error was not caused by a bit flip in an FPGA, and thus is not an FSEU (i.e., a non-FSEU).

Analyzing the collected data, it was found that a significant percentage of the total cases studied could not be classified as either an FSEU or non-FSEU. For the majority of the indetermi-

Authorized licensed use limited to: Northeastern University. Downloaded on April 21, 2009 at 09:19 from IEEE Xplore. Restrictions apply.

TABLE I
PROBABLE FSEU ERROR CODE STATISTICS

Probable FSEU Error Code Stats	Percentage
FSEU	29.35
Non-FSEU	33.20
Indeterminate	37.45

TABLE II
POTENTIAL FSEU ERROR CODE STATISTICS

Potential FSEU Error Code Stats	Percentage
FSEU	14.45
Non-FSEU	40.46
Indeterminate	45.09

TABLE III
FSEU DISTRIBUTION IN FPGA MODULES

FSEU Location	Percentage
FPGA-1	9.5%
FPGA-2	59.5%
FPGA-1 or FPGA-2	31.0%

nate cases, our analysis is incomplete due to the lack of an error log file being available, and thus these errors could not be properly analyzed to the level of detail needed. However, by using the trends observed in the FSEU/non-FSEU errors, we can approximate what portion of the indeterminate cases are FSEUs and non-FSEUs. We multiply the ratio of the FSEUs/(FSEUs + non-FSEUs) to provide us with an estimate of the number of indeterminate cases that are FSEUs.

Table I represents all errors that were found to have a *Probable FSEU* error code. The FSEU category represents the errors whose log file had an FSEU failure signature; the non-FSEU category represents the errors whose log file did not have an FSEU failure signature. The indeterminate category represents the errors whose log file was not available, and thus whose FSEU error status could not be determined. Table II shows this same information for all errors that contained *Potential FSEU* error codes.

The next step in calculating the FIT rate of the targeted FPGA was to determine exactly where each FSEU occurred relative to the LUM. By again referencing each FSEU's field returned error code, information detailing the location of each FSEU could be extracted. Here, the FSEUs were categorized as having occurred on FPGA-1, on FPGA-2, or on either FPGA-1 or FPGA-2. The breakdown is shown in Table III.

Assuming that FPGA-1 is the component of interest, and also that the distribution of FPGA-1 FSEUs within the category of FPGA-1 or FPGA-2 ranges from 20% to 80%, the following calculations result:

If 20% of these errors were to occur on FPGA-1, then the total number of FSEUs on FPGA-1 was calculated by:

$$(0.095 + (0.20 \times 0.31)) \times (\text{The total number of FSEUs}).$$

Similarly, if 80% of these errors were to occur on FPGA-1, then the total number of FSEUs on FPGA-1 was calculated by:

$$(0.095 + (0.80 \times 0.31)) \times (\text{The total number of FSEUs}).$$

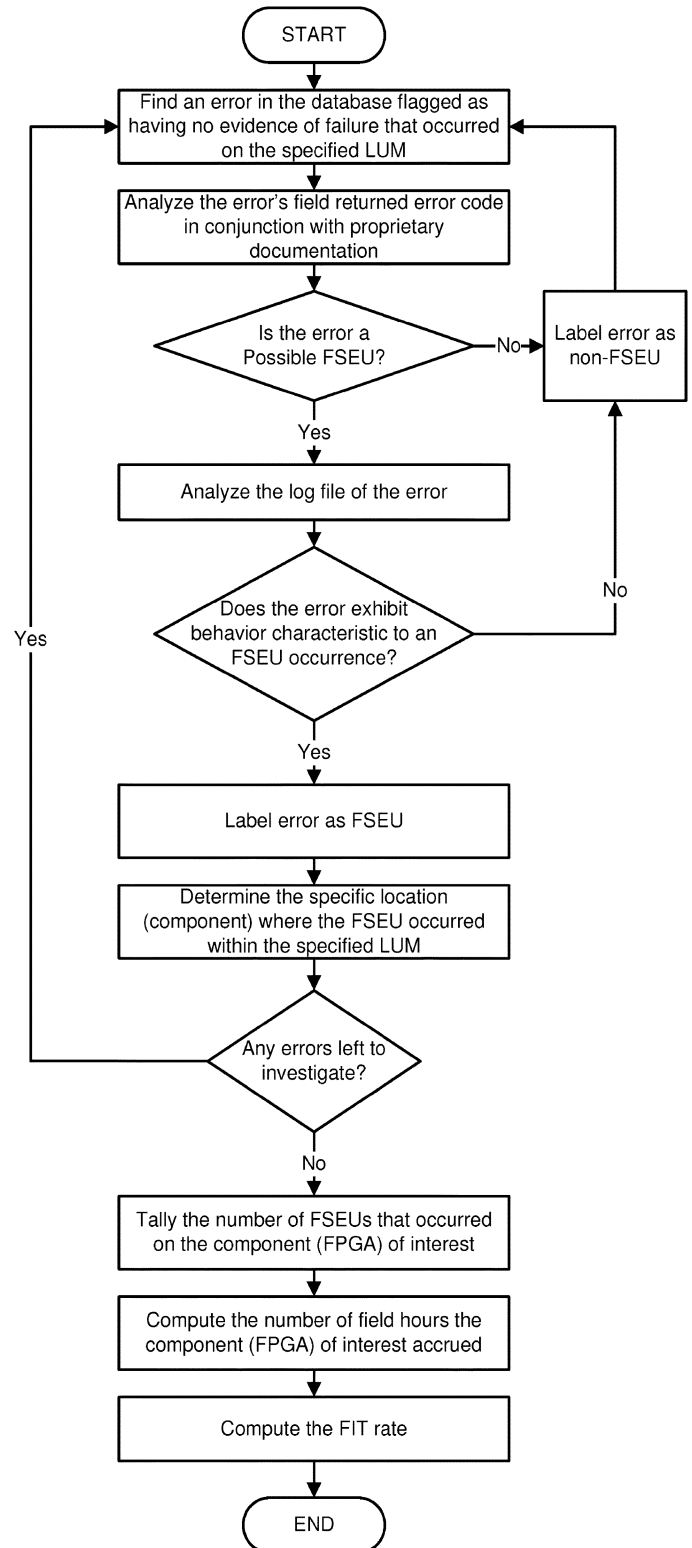

Fig. 5. Field data analysis flow.

The final step of this project was to then calculate an overall FIT rate for the FPGA of interest, in this case FPGA-1. By analyzing the number of FSEUs that occurred on this FPGA in combination with the total number of field hours that this component accrued, a device field FIT rate was obtained. Fig. 5 summarizes all of the steps taken in computing this value.

Authorized licensed use limited to: Northeastern University. Downloaded on April 21, 2009 at 09:19 from IEEE Xplore. Restrictions apply.

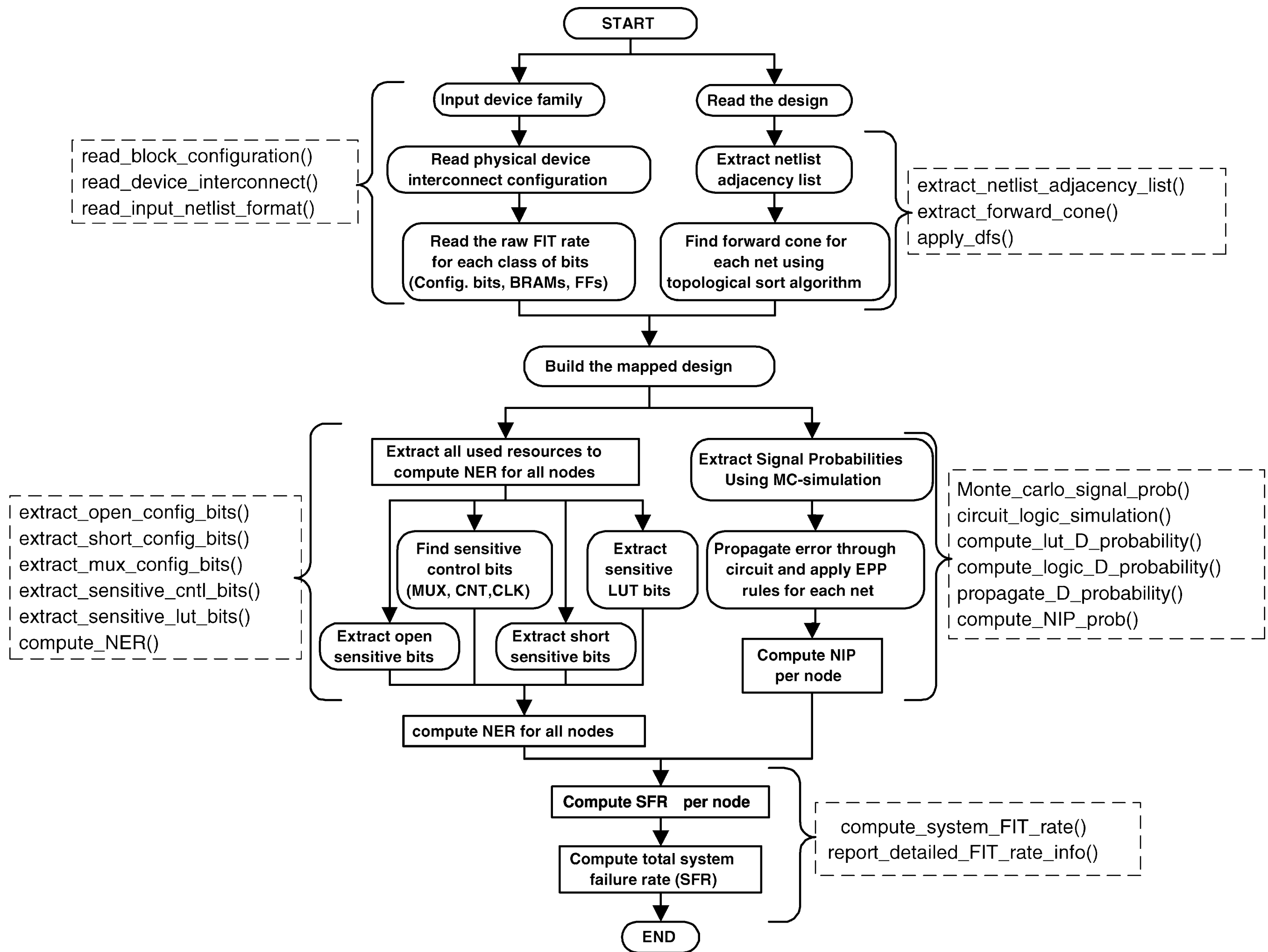

Fig. 6. SER estimation flow.

C. Tool Implementation

Fig. 6 shows the overall flow of the presented SER estimation methodology. We have developed a fully automated software tool implemented in Object Oriented Programming (OOP) using C++ language. This tool automatically performs all the steps shown in Fig. 6. The inputs to the program are physical device information, interconnect and architecture information, and the raw error rates for different types of cells. Our framework produces the following outputs: a) system overall FIT rate, b) system overall raw FIT rate, c) detailed FIT rate for each net, logic block, look-up table, routing and clock/control resources.

There are four categories of functions in the framework listed below:

i) Device Physical Information Modules.
ii) Design Information Modules: these modules hold design information.
iii) Used Resources Information: these modules extract the used resource information.
iv) System Failure Computation: these modules compute the system FIT rate.

Among the four categories of modules, only the first category is needed to be updated for each FPGA device family and architecture while the three last categories remain unchanged. This can help to minimize code maintenance cost. These four categories of modules are shown in Fig. 6.

D. Generalization to Other FPGA Architectures

The proposed methodology can be easily extended to other SRAM-based FPGA architectures. In fact, the main analytical modeling technique is independent of the underlying FPGA physical architecture. This is because this SER analysis methodology is composed of two major parts. In the first part, we abstract the FPGA resource utilization for the mapped design by *Node Error Rate* factor. This part is dependent on the FPGA architecture. The second part, which is the major part of this methodology, performs soft error analysis at the gate-level, which is completely independent of the FPGA architecture. The way that the tool is implemented is completely compatible with this idea, and the implementation of these two parts are completely independent. Moreover, the tool is designed in such a way that it accepts various implementations and modifications for the first part to incorporate different FPGA architectures.

Authorized licensed use limited to: Northeastern University. Downloaded on April 21, 2009 at 09:19 from IEEE Xplore. Restrictions apply.

TABLE IV
FPGA RESOURCE UTILIZATION INFORMATION FOR ECU CHIP

Parameter	Slices	IOBs	BRAM bits	LUTs	FFs	Configuration bits
Number Used	12286	631	290816	18033	9725	1.7M
Total	12288	728	393216	24576	24576	6.5M
Usage	99.9%	86.7%	73.9%	73.4%	40.0%	26.2%

TABLE V
EXECUTION TIME FOR ECU SER ESTIMATION

Parameter	Read	Arrange	SP Computation	Extract error sites	SFR calculation	Total Time
Time (second)	108.74	3.85	437.30	50.99	11780.00	**12380.88**
Percentage	0.88%	0.03%	3.53%	0.41%	95.15%	**100.00%**

Note that the main differences in various FPGA architectures are the programmable interconnect networks. Some of the architectures are based on programmable multiplexers (such as Altera FPGAs) while others are based on programmable switch matrices (such as Xilinx FPGAs). The differences in the logic blocks are typically minimal. As we described in previous subsection, to apply this SER methodology to any FPGA architecture, only the device physical information is needed to be updated in the tool to reflect the corresponding FPGA architecture. This mainly captures the number of sensitive bits (configuration SRAM cells) associated with each node and net in the gate-level netlist of the design mapped into the FPGA device. Such information can be obtained from the output of FPGA placement and routing phases. The device physical information modules need to be extended to extract sensitive bit information from the final placement and routing results for each FPGA architecture and device family. Once this information provided, the system failure rate is computed using the netlist impact probability and the node error rates.

E. Analytical Results

In order to validate our proposed SER estimation methodology, we have studied the embedded control unit (ECU) in the information system. The ECU design has been implemented using a very popular commercial FPGA device. Table IV shows the FPGA utilization of this device for the ECU design. As seen in this table, the ECU design uses 99.9% of total slices and 73.4% of total LUTs.

The error list considered in the experiments includes mux-open, PIP open, PIP short, buffer-on, buffer-off, LUT bit-flip, and control/clocking bit-flip, and bridging errors. The implemented software tool extracts the netlist information from the input 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 Sun Blade 900 © workstation equipped with 4 GB main memory and running Solaris 9 © operating system.

The sensitive bits are classified according to the error models described in Section III. According to these results, the configuration routing bits constitute almost 85% of the total sensitive configuration bits while LUT bits and control/clocking bits constitute 11% and 4% of total configuration bits, respectively. The results also show that the number of FFs is less than 0.6% of the number of sensitive configuration bits.

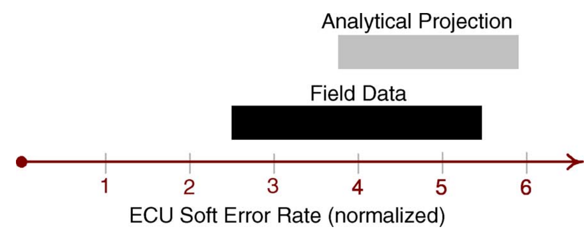

Fig. 7. Comparison of predictive tool results with field data.

The detailed execution time of our SER estimation method is listed in Table V. The total run time of this SER estimation method includes the time required to:

- read the netlist;
- arrange the netlist;
- compute signal probabilities;
- extract error sites;
- compute error propagation probability including the time needed to compute NER_i and NIP_i for all nodes.

One of the main advantages of the proposed SER estimation technique is that detailed FIT rate information per net, per used resource, and per resource type can be extracted. These detailed FIT rate reports can be very beneficial to designers for soft error debug and diagnosability. Using these detailed FIT rates, designers can identify the most vulnerable paths, modules, and components and prioritize them according to their FIT rates. Based on the FIT rate rankings, selective protection schemes can be applied to more susceptible paths and components.

The system failure rate of the ECU has been computed using both our predictive tool and the collected field data. Fig. 7 shows the comparison of our tool results for the ECU and the ECU field data. This figure shows normalized numbers. Based on the raw FIT rates reported by vendors, our predictive tool reports a FIT rate ranging from 3.8 to 5.9. Also, the field data analysis shows that the ECU FIT rate ranges from 2.5 to 5.5.

The slight mismatch between the field data and our analytical projection can be attributed to two reasons:

- First, it was not possible to catch all FSEUs that occurred on the investigated device in the field. This is primarily due to the fact that some FSEUs do not propagate to system outputs and remain undetected. Also, our list of *Possible FSEUs* does not cover all FSEUs and is continually being updated to try and do so.

Authorized licensed use limited to: Northeastern University. Downloaded on April 21, 2009 at 09:19 from IEEE Xplore. Restrictions apply.

TABLE VI
MODULAR SUSCEPTIBILITY ANALYSIS

Module name	FIT(%)	% used SRAM cells
module_1	18.51	25.26
module_2	9.73	3.25
module_3	9.32	13.73
module_4	8.87	11.30
module_5	5.85	1.50
module_6	5.73	6.78
module_7	3.47	3.06
module_8	2.33	3.88
module_9	2.32	3.86
module_10	2.30	3.93
the rest	**31.57**	**23.45**

- Second, this discrepancy can be partially attributed to some potential inaccuracies in the raw fit rates as published by FPGA vendors.

Note that it is expected that the proposed analytical SER methodology produces more accurate results for applications in which accurate raw FIT rates are provided.

We have also analyzed the susceptibility of each architectural module within the ECU. Using this analysis tool it is possible to investigate the soft error vulnerability of each component in the hierarchical architecture from top-level modules down to a net or cell within a sub-module. Such information is very useful for soft error debug, diagnosis, and re-design. The FIT contribution and the percentage of used SRAM cells of each top-level module, as an indication of the relative size of each module, is reported in Table VI. As can be seen from this table, the soft error contributions of different architectural modules are not uniform. Furthermore, some smaller modules (i.e., mapped using a small fraction of the FPGA resources) contribute to a considerable percentage to the overall FIT rate. Such information can be used by architects and designers to include protection (in terms of architectural or device redundancy) for those modules.

VI. CONCLUSIONS

Dependability is one of the most critical factors for information systems. Since FPGAs are commonly used in the implementation of these systems (particularly in the design of Logical Unit Modules) it is important to be able to estimate the soft error reliability of FPGA-based designs. Designs mapped into FPGAs are more susceptible to soft errors than ASIC implementations since the majority of an FPGA chip area is dedicated to memory elements storing the configuration of the FPGA or circuit state. Moreover, soft errors in configuration memory cause permanent errors in the mapped design which cannot be corrected by traditional retry mechanisms.

In this paper, we described an efficient and accurate SER estimation framework for FPGA-based designs. Besides its accuracy and speed, detailed FIT rate information produced by our toolset can be effectively used for soft error debugging and diagnosis which can help to improve system reliability with minimal cost and performance overheads.

We have also presented a case study on an FPGA-based controller (implemented on the largest FPGA device) used in the design of a commercial information system. We have compared FIT rate obtained from comprehensive field data analysis versus results produced from our framework. These results show that our analytical framework can accurately predict FIT rates (there is an 81% overlap in FIT rates obtained between the analytical mode and the field failure data) while the runtime of our framework is completely tractable (3.5 hours for a very large design mapped into one of the largest commercial FPGA devices).

REFERENCES

[1] M. Alderighi, F. Casini, S. D. Angelo, M. Mancini, A. Marmo, S. Pastore, and G. R. Sechi, "A tool for injecting SEU-like faults into the configuration control mechanism of Xilinx Virtex FPGAs," in *Proc. 18th IEEE Symp. Defect and Fault Tolerance in VLSI Systems*, 2003, pp. 71–78.

[2] G. Asadi, G. Miremadi, H. R. Zarandi, and A. Ejlali, "Fault injection into SRAM-based FPGAs for the analysis of SEU effects," in *Proc. IEEE Int. Conf. Field-Programmable Technology*, Tokyo, Japan, Dec. 2003, pp. 428–430.

[3] G. Asadi and M. B. Tahoori, "An accurate SER estimation method based on propagation probability," in *Proc. IEEE/ACM Int. Conf. Design, Automation and Test in Europe*, Munich, Germany, Mar. 2005, pp. 306–307.

[4] G. Asadi and M. B. Tahoori, "An analytical approach for soft error rate estimation in digital circuits," in *Proc. IEEE Int. Symp. Circuits and Systems*, Kobe, Japan, May 2005, vol. 3, pp. 2991–2994.

[5] G. Asadi and M. B. Tahoori, "Soft error rate estimation and mitigation for SRAM-based FPGAs," in *Proc. 13th ACM Int. Symp. Field-Programmable Gate Arrays*, Monterey, CA, Feb. 2005, pp. 149–160.

[6] R. C. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies," *IEEE Trans. Device Mater. Rel.*, vol. 5, no. 3, pp. 305–316, Sep. 2005.

[7] M. Bellato, M. Ceschia, M. Menichelli, A. Papi, J. Wyss, and A. Paccagnella, "Ion beam testing of SRAM-based FPGAs," in *Proc. 6th European Conf. Radiation and Its Effects Components and Systems*, Sep. 2001, pp. 474–480.

[8] C. Carmichael, E. Fuller, J. Fabula, and F. Lima, "Proton testing of SEU mitigation methods for the Virtex FPGA," in *Proc. Military and Aerospace Applications Programmable Logic Devices*, Washington, D.C., Sep. 2001.

[9] M. Ceschia, M. Bellato, A. Paccagnella, and A. Kaminski, "Ion beam testing of ALTERA APEX FPGAs," in *Proc. IEEE Radiation Effects Data Workshop*, Jul. 2002, pp. 45–50.

[10] M. Ceschia, M. Violante, M. S. Reorda, A. Paccagnella, P. Bernardi, M. Rebaudengo, D. Bortolato, M. Bellato, P. Zambolin, and A. Candelori, "Identification and classification of single-event upsets in the configuration memory of SRAM-based FPGAs," *IEEE Trans. Nucl. Sci.*, vol. 50, no. 6, pp. 2088–2094, Dec. 2003.

[11] 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," presented at the IEEE Nuclear and Space Radiation Effects Conf., Reno, NV, Jul. 2000.

[12] M. Gokhale, P. Graham, E. Johnson, N. Rollins, and M. Wirthlin, "Dynamic reconfiguration for management of radiation-induced faults in FPGAs," in *Proc. 18th Int. Parallel and Distributed Processing Symp.*, Santa Fe, NM, Apr. 2004, pp. 145–150.

[13] P. Graham, M. Caffrey, J. Zimmerman, D. E. Johnson, P. Sundararajan, and C. Patterson, "Consequences and categories of SRAM FPGA configuration SEUs," presented at the Military and Aerospace Applications Programmable Logic Devices Int. Conf., Washington, DC, Sep. 2003.

[14] 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," in *Proc. Symp. VLSI Technology*, Jun. 2001, Digest of Technical Papers, pp. 73–74.

[15] Hennessy and Patternson, *Computer Architecture: A Quantitative Approach*, 3rd ed. San Mateo, CA: Morgan Kaufmann, 2003.

[16] E. Johnson, M. Caffrey, P. Graham, N. Rollins, and M. Wirthlin, "Accelerator validation of an FPGA SEU simulator," *IEEE Trans. Nucl. Sci.*, vol. 50, no. 6, pp. 2147–2157, Dec. 2003.

[17] J. Karlsson, P. Ledan, P. Dahlgren, and R. Johansson, "Using heavy-ion radiation to validate fault handling mechanisms," *IEEE Micro*, vol. 14, no. 1, pp. 8–23, Feb. 1994.

[18] T. Karnik, B. Bloechel, K. Soumyanath, V. De, and S. Borkar, "Scaling trends of cosmic rays induced soft errors in static latches beyond 0.18 μ," in *Proc. Symp. VLSI Circuits*, Jun. 2001, Digest of Technical Papers, pp. 61–62.

Authorized licensed use limited to: Northeastern University. Downloaded on April 21, 2009 at 09:19 from IEEE Xplore. Restrictions apply.

[19] R. Katz, K. LaBel, J. J. Wang, B. Cronquist, R. Koga, S. Penzin, and G. Swift, "Radiation effects on current field programmable technologies," *IEEE Trans. Nucl. Sci.*, vol. 44, no. 6, pp. 1945–1956, Dec. 1997.

[20] A. Lesea, S. Drimer, J. J. Fabula, C. Carmichael, and P. Alfke, "The rosetta experiment: Atmospheric soft error rate testing in differing technology FPGAs," *IEEE Trans. Device Mater. Rel.*, vol. 5, pp. 317–328, Sep. 2005.

[21] A. Lesea and J. J. Fabula, "The Rosetta experiment: Atmospheric soft error rate testing in differing technology FPGAs – 90 nanometer update," presented at the System Effects of Logic Soft Errors II Workshop, Urbana-Champaign, IL, Apr. 2006.

[22] P. Liden, P. Dahlgren, R. Johansson, and J. Karlsson, "On latching probability of particle induced transients in combinational networks," in *Proc. 24th Symp. Fault-Tolerant Computing*, 1994, pp. 340–349.

[23] F. Lima, C. Carmichael, J. Fabula, R. Padovani, and R. Reis, "A fault injection analysis of Virtex FPGA TMR design methodology," in *Proc. 6th European Conf. Radiation Effects Components and Systems*, Grenoble, France, 2001, pp. 275–282.

[24] S. Mitra, N. Seifert, M. Zhang, Q. Shi, and K. Kim, "Robust system design with built-in soft-error resilience," *IEEE Computer*, vol. 38, pp. 43–52, Feb. 2005.

[25] K. Mohanram and N. A. Touba, "Cost-effective approach for reducing soft error failure rate in logic circuits," in *Proc. Int. Test Conf.*, Charlotte, NC, Oct. 2003, pp. 893–901.

[26] K. Morgan, M. Caffrey, P. Graham, E. Johnson, B. Pratt, and M. Wirthlin, "SEU-induced persistent error propagation in FPGAs," *IEEE Trans. Nucl. Sci.*, vol. 52, no. 6, pp. 2438–2445, Dec. 2005.

[27] S. S. Mukherjee, C. Weaver, J. Emer, S. K. Reinhardt, and T. Austin, "A systematic methodology to compute the architectural vulnerability factors for a high-performance microprocessor," in *Proc. 36th Int. Symp. Micro-Architecture*, San Diego, CA, 2003, pp. 29–40.

[28] B. Mullins, H. Asadi, M. B. Tahoori, D. Kaeli, K. Granlund, R. Bauer, and S. Romano, "Case study: Soft error rate analysis in storage systems," in *Proc. IEEE VLSI Test Symp.*, Berkeley, CA, May 2007, pp. 256–264.

[29] H. T. Nguyen and Y. Yagil, "A systematic approach to SER estimation and solutions," in *Proc. 41st Int. Reliability Physical Symp.*, Dallas, TX, 2003, pp. 60–70.

[30] E. Normand, "Single event upset at ground level," *IEEE Trans. Nucl. Sci.*, vol. 43, no. 6, pp. 2742–2750, Dec. 1996.

[31] K. P. Parker and E. J. McCluskey, "Probabilistic treatment of general combinational networks," *IEEE Trans. Comput.*, vol. C-24, no. 6, pp. 668–670, Jun. 1975.

[32] M. Rebaudengo, M. S. Reorda, and M. Violante, "Simulation-based analysis of SEU effects on SRAM-based FPGAs," in *Proc. 12th Int. Conf. Field-Programmable Logic and Applications*, Montpellier, France, Sep. 2002, pp. 607–615.

[33] M. S. Reorda, L. Sterpone, and M. Violante, "Efficient estimation of SEU effects in SRAM-based FPGAs," in *Proc. 11th IEEE Int. On-Line Testing Symp.*, Saint Raphael, French Riviera,, France, Jul. 2005, pp. 54–59.

[34] P. Shivakumar, M. Kistler, S. W. Keckler, D. Burger, and L. Alvisi, "Modeling the effect of technology trends on the soft error rate of combinational logic," in *Proc. Int. Conf. Dependable Systems and Networks*, Washington, D.C., Jun. 2002, pp. 389–399.

[35] L. Sterpone and M. Violante, "A new analytical approach to estimate the effects of SEUs in TMR architectures implemented through SRAM-based FPGAs," *IEEE Trans. Nucl. Sci.*, vol. 52, no. 6, pp. 2217–2223, Dec. 2005.

[36] G. M. Swift, S. Rezgui, J. George, C. Carmichael, M. Napier, J. Maksymowicz, J. Moore, A. Lesea, R. Koga, and T. F. Wrobel, "Dynamic testing of Xilinx Virtex-II field programmable gate array (FPGA) input/output blocks (IOBs)," *IEEE Trans. Nucl. Sci.*, vol. 51, no. 6, pp. 3469–3474, Dec. 2004.

[37] M. B. Tahoori, S. Mitra, S. Toutounchi, and E. J. McCluskey, "Fault grading FPGA interconnect test configurations," in *Proc. Int. Test Conf.*, Baltimore, MD, Oct. 2002, pp. 608–617.

[38] M. Violante, L. Sterpone, M. Ceschia, D. Bortolato, P. Bernardi, M. S. Reorda, and A. Paccagnella, "Simulation-based analysis of SEU effects in SRAM-based FPGAs," *IEEE Trans. Nucl. Sci.*, vol. 51, no. 6, pp. 3354–3359, Dec. 2004.

[39] M. Wirthlin, E. Johnson, N. Rollins, M. Caffrey, and P. Graham, "The reliability of FPGA circuit designs in the presence of radiation induced configuration upsets," in *Proc. IEEE Symp. Field-Programmable Custom Computing Machines*, Napa, CA, Apr. 2003, pp. 133–142.

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

[41] Virtex-II 1.5 V Field-Programmable Gate Arrays Xilinx, San Jose, CA, 2001, Data Sheet DS031-1 (v1.7).

[42] J. F. Ziegler, "Terrestrial cosmic rays," *IBM J. Res. Develop.*, vol. 40, no. 1, pp. 19–39, Jan. 1996.

Authorized licensed use limited to: Northeastern University. Downloaded on April 21, 2009 at 09:19 from IEEE Xplore. Restrictions apply.