Parallel SER Analysis for Combinational and Sequential Standard Cell Circuits[☆]

Weiguang Sheng*, Jianfei Jiang*, Zhigang Mao*

*Department of Micro/Nano Electronics, Shanghai Jiao Tong University, Shanghai
200240, China*

Abstract

A parallel SER (soft error rate) evaluation framework ASSET-VLG was developed to analyze the SER of both combinational and sequential standard cell circuits. ASSET-VLG was constructed in practically oriented way: i) it employs a verilog parser for automatically reading the synthesized DUT (device under test) netlist; ii) it provides an accurate and unified SER analysis framework for both the combinational and sequential circuits rather than the former only; iii) it targets to a 130 nm production library and the modeling method can be easily ported to newer technologies. Furthermore, concurrency is also exploited for accelerating the evaluation procedure on modern multi-core computers. These features make ASSET-VLG appropriate for automatic SER estimation in design stage and can be easily integrated into current highly reliable ICs design flow. Experiments on ISCAS'85 and ISCAS'89 benchmark circuits show the evaluation time ranges from 0.5 ms to 2.16 s without previous memory explosion problem. Compared with spice, the modeling method of ASSET-VLG provides 98% accuracy. The parallelizing experiments indicate the proposed method has better scalability, e.g., 4.44X speedup are obtained in a 4 cores/8 threads platform. The experiments also reveal that sequential part (flip-flops) in the circuit dominating the system SER by one order than combinational gates for a 130 nm CMOS

[☆]This work was supported by National Science Foundation of China under grant No. 61201059. The author would also thank the partial support from the National High Technology Research and Development Program of China (No. 2012AA012702).

*Corresponding author

Email addresses: wgshenghit@gmail.com (Weiguang Sheng),
jiangjianfei@sjtu.edu.cn (Jianfei Jiang), maozhigang@sjtu.edu.cn (Zhigang Mao)

process. Last but not least, significant frequency dependence of SER are observed in flip-flops, implying the commonly used critical charge measure is insufficient for characterizing soft error in sequential cells.

Keywords: Soft Error, SER, Combinational, Sequential, Parallel, Standard Cell

1. Introduction

Soft error due to high-energy particles striking has been a main challenge for modern deep sub-micron CMOS ICs. The particles include α particle emitted from the radioactive impurities in package materials, and neutron or proton come from high-energy cosmic rays. As technology scaling, even the harmless muon in the past has become a newer soft error source in nanoscale circuits [1].

Soft error seriously affects the reliability of ICs, while robust circuit design requires solutions to the accurate, automatic and systematic SER (soft error rate) estimation methodologies [2]. Soft error rate has various metrics: in logical level, FIT (failure in time, one failure per billion device hours) is used to measure the circuit SER [3]; while for storage elements, cross section or critical charge are used instead [4]. The most straightforward SER estimation method is to use particle accelerator or heavy-ion equipment to bombard the DUT (device under test) [5]. However, the bombarding approach is expensive and only applicable when chip has been fabricated. Designers demand distinctive SER evaluation tools designated for design stage.

In the past decades, intensive researches have been carried out for characterizing circuit SER in design stage to mitigate the gap between circuit design and reliability evaluation [6–11]. These works can help us estimate SER in design stage promptly and avoid the high reengineering cost after taping out for reliability issue. SER evaluation tool has been a key factor for highly reliable circuit design flow.

Though great advancements have been achieved, SER evaluation is still an open research problem, especially in making the methodologies more systematic, automatic, fast and industry-oriented [2]. We developed a new tool ASSET-VLG (Advanced Statistical SER Evaluation Toolkit for Verilog) to address the aforementioned problems. ASSET-VLG contains a bunch of programs developed using C++ and Python language, which reads in DUT's gate level netlist generated by logic synthesizer, and performs SER estimation

fully automatically.

The contributions of this paper can be summarized as follows:

1. It provides a systematic methodology to analyze SER of various circuits. Both combinational and sequential logic can be analyzed by ASSET-VLG, while many previous researches are designated for only one specific type circuits.
2. The entire SER evaluation flow of ASSET-VLG is fully automated by various accompanied automation tools: library modeling, input vector preparation, SER analyzer, etc.
3. LUT (lookup table) with trilinear interpolation approach is utilized to model the single event glitch generation and propagation, providing at least 98% modeling accuracy compared to spice simulator.
4. As known to us, it might be the first time multithreaded parallel programming techniques were introduced to the SER analysis realm by utilizing the power of modern multi-core platform. Although [9] had exploited the bit-level parallelism (at the cost of ignoring electrical masking) emerged in the logic simulation field.
5. A fast frequency aware SER analysis tool for flip-flops was developed by combining statistical fault injection and curve fitting techniques. By using this tool, we found there are dramatic frequency dependence of flip-flop SER. The same phenomenon was also investigated by a recent work utilizing a different method and technology [12].
6. With the help of ASSET-VLG, we found that sequential logic (flip-flops) dominating the SER of 130 nm benchmark circuits. Similar results was also obtained by a circuit level SER estimation tool ASSET-SPI developed by us on the 180 nm technology, which reminds us the main soft error challenge before 130 nm technology still comes from the sequential cells rather than the combinational part predicated in other works.

The rest of this paper are organized as follows: Section 2 reviews the backgrounds and related works; Section 3 describes the theoretical framework of ASSET-VLG; Section 4 detailedly describes the implementation of ASSET-VLG and solutions to some key problems; Section 5 gives experimental results and the discussions on the results; Section 6 concludes the whole work.

2. Related Works

In circuits, particles striking the reverse biased transistor will generate a small voltage glitch (SET, single event transient) at the drain node. The glitch might be latched by a succeeded storage elements (latch or flip-flop, etc.) and destroy its original value, thus a SEU (single event upset) or soft error occurs. As technology scaling and operating frequency increase, SET originating from combinational logic has much more chances to get latched by storage cells and upsetting the chip. It is believed that for the technology beyond 100 nm, SER from combinational logic will exceed the sequential logic and dominate the chip SER [4, 13]. However, recent researches in [14, 15] show that sequential logic soft errors still dominate at features sizes as small as 32 nm, e.g., for a 32nm bulk CMOS process, the SER of one single logic chain is less than 10% of the nominal latch SER at 1 GHz [14]. Therefore, both the combinational and sequential logic should be taken into account in the comprehensive SER analysis.

Researches in [16, 17] employed Monte Carlo based technique to model soft error from physical perspective with too many lower details, which made the technique time-consuming and applicable for device level modeling rather than higher level large-scale SER analysis. Simulated fault injection technique can also be used to evaluate chip SER by injecting soft error into DUT's simulation model [18, 19]. However, fault injection has two disadvantages: i) it requires large number of injections to obtain statistical unbiased results; ii) simplified bit-flip model used in fault injection loses the glitch width and amplitude details, which makes it more appropriate for gross SER estimation at the behavioral level.

Analytical SER evaluation methodologies make good trade-off between accuracy and resource requirements. It models the SET glitch generation and propagation by analytical approach to obtain the SER estimation of DUT. Generally spice results are utilized as the golden standard to evaluate the accuracy of analytical methodologies. For SET glitch generation, Figure 1 shows two SET current models mostly adopted in analytical approaches, where one is for gate level modeling and the other is for more detailed transistor level modeling. In Figure 1(a), a current source is attached at the gate output to model the SET glitch generation, while in Figure 1(b) the current source is inserted between the drain and the body of the transistor [20]. The latter is more close to physical reality because any transistor in gate may get struck but Figure 1(a) assumes only gate output is struck by particles.

Figure 1: Current models for SET glitch generation

Generated SET glitches will propagate in the circuit and may be latched by storage elements. The propagation and latching of SET are affected by three masking mechanisms [13]: i) logical masking - there must exist a sensitive path from the striking site to the primary output for the SET propagation, otherwise the glitch will be filtered out; ii) electrical masking - SET without enough amplitude or width will be attenuated by gate in the data path and evaporate eventually; iii) latching-window masking - SET must reach the flip-flop in legal latching window to be latched successfully. Various methods have been proposed to model the SET glitch generation and propagation to estimate the SER by analytical approach.

ASERTA in [21] used fault simulation and heuristic rules to compute the logical and electrical masking probability to estimate the soft-error tolerance of nanometer combinational circuits. In [10, 22], complicated analytical equations were employed to model the SET propagation to storage cells, and then compute the combinational logic SER based on fault simulation. [8, 23, 24] used symbolic techniques and decision diagrams to analyze combinational logic SER, but the memory explosion problem restricts these techniques from handling larger benchmark circuits. [9] used novel signature based technique to compute SER. An important improvement of [9] is exploiting the bit-level parallelism that commonly used in logical simulation, but the electrical masking is hard to be considered by this approach. [7] is another comprehensive SER estimation work, which employed novel Weibull function to represent the SET waveform. Moreover, the state and node dependence problem in cell characterization, along with the important waveforms refinement problem in glitch propagation were also considered in [7]. [25] employed support

vector regression technique for the glitch generation and propagation calculation, but the machine learning approach has higher computing complexity and limit the analysis speed. CASSER in [11] used statistical closed-form analysis technique to examine the SER fluctuation under process variation, and reported dramatic performance improvement over reference [25].

As aforementioned, sequential logic soft error is still critical in whole chip SER [14, 15]. [26] addressed the SER estimation of ISCAS'89 sequential benchmark circuits by calculating the error propagation probability. [27] used time frame expansion method to calculate the sequential circuit SER after multi-cycles glitch propagation. [28] calculated the SER of sequential circuits based on symbolic analysis and multi-cycles circuit unrolling. However, these works estimated the SER only for the combinational core of sequential circuits rather than whole chip including DFFs (D flip-flop). [29] figured out the sequential logic SER should be taken into account, but it only let the sequential logic SER to zero to simplify the SER calculation. [3] gave a chip level SER estimation flow covering static gates, domino gate, latch etc., while logic masking and latching-window masking were considered but electrical masking had not been mentioned. [6] constructed a comprehensive theory framework for SER estimation and proposed the SERA tool based on probability theory, circuit simulation, graph theory, and fault simulation. SERA divides a DFF into sample stage and hold stage and then calculates the two stages SER separately. Therefore SERA covers both combinational logic and DFF SER, but the simplified equivalent inverter chains model might restrict its accuracy.

3. SER Estimation Methodology

3.1. System SER Analysis

ASSET-VLG in this work can handle logic circuits containing both combinational logic and sequential logic. For any logic circuit being analyzed, ASSET-VLG partitions it into combinational part and sequential part. The combinational part denotes the dash line box in Figure 2 (except the DFF in elliptic box) composed of static gates, while sequential part refers to the components (mainly DFFs) other than static gates. The partitioned circuit contains a pure combinational logic and a sequential DFFs list, and the SER of these two parts are evaluated separately. The partitioning procedure is executed automatically according to the following rules.

1. put all primary inputs (PI) to PI list;

2. put all primary outputs (PO) to PO list;
3. put all DFFs (include internal DFFs) to DFF list;
4. attach each unregistered primary output with a virtual basic size DFFX1;
5. for all internal DFFs, put their D nets to PO list, and Q nets to PI list.

Figure 2: Partitioning circuit into combinational and sequential part

A soft error occurs when a glitch originated at combinational logic is latched by a DFF, or the DFF itself in sequential logic is upset by particle striking. Therefore the SER in this work is only nominal SER. As a SER measure, FIT is superior to other measures such as architectural vulnerability factor for its additivity [3, 30]. Hence the nominal system SER can be obtained by summing up the SER of combinational part and sequential part, as described in Equation(1).

$$SER_{sys} = SER_{com} + SER_{seq}, \quad (1)$$

where SER_{sys} , SER_{com} , SER_{seq} refer to the whole circuit SER, combinational SER and sequential SER respectively.

As an example, the circuit in Figure 2 has two primary inputs X1 and X2, and two primary outputs Y1 and Y2. O1 is a DFF at primary output Y1; V1 is a virtual DFF append by ASSET-VLG at Y2 and Y2' is a newly added primary output for Y2. Furthermore, the internal DFF I1 is categorized to sequential part, while its input Y3' is treated as a new primary output and the output X3' is treated as a new primary input. Thus the circuit is partitioned into combinational part (G0-G4) and sequential part (3 DFFs I1, O1, V1), and the whole SER is obtained by summing up the two parts SER as depicted in Equation(1).

3.2. Combinational Logic SER

The SER calculation methodology here for combinational logic is derived from [6], although distinctive circuit partition method has been employed. For a circuit with N nodes, the system SER satisfies the following inequality:

$$SER \leq \sum_{i=1}^N SER_i, \quad (2)$$

where SER_i is the SER that node i be struck and upsets the circuit. The inequality becomes an equality when the N nodes are independent. The charge collected varies in the range $[Q_{min}, Q_{max}]$ depending on impacted particle type, energy, angle, etc., thus Equation (2) is transformed to an expectation form:

$$SER \leq \sum_{i=1}^N \int_{q=Q_{min}}^{Q_{max}} SER_i(q) dq, \quad (3)$$

where $SER_i(q)$ is the probability density function for node i collecting charge q and upset the circuit. $SER_i(q)$ is calculated by the following two equations:

$$SER_i(q) = R_i(q) \cdot P_i(q), \quad (4)$$

$$R_i(q) = F \cdot k \cdot A_i \cdot \frac{1}{Q_s} \cdot e^{-\frac{q}{Q_s}}. \quad (5)$$

$R_i(q)$ in Equation(4) is the striking rate that node i being struck and collecting q charge. $R_i(q)$ is calculated by Equation(5), where F is the neutron flux depending on the location and altitude; k is the fraction of particle hits resulting in charge generation [31]; A_i is the sensitive area (drain diffusion area of transistor) of node i ; Q_s is charge collection slope; q is the charge collected by particle striking.

$P_i(q)$ in Equation(4) is the probability node i (collects charge q) upsets the circuit (any output terminal captures a erroneous value). The calculation of $P_i(q)$ is the key problem in any SER estimation research, and we estimate the its upper bound by the following inequality:

$$P_i(q) \leq \sum_{j=1}^O P_i^j(q), \quad (6)$$

where O is the number of output terminals number and $P_i^j(q)$ is the probability a particle hitting node i , collecting charge q and arousing erroneous value capturing in output terminal j . The inequality holds because one hitting might upset multiple output terminals. By combining the previous equations, the upper bound of combinational logic SER is estimated by:

$$SER_{comb} = F \cdot k \cdot \sum_{i=1}^N \sum_{j=1}^O \int_{Q_{min}}^{Q_{max}} A_i \cdot \frac{1}{Q_s} \cdot e^{-\frac{q}{Q_s}} \cdot P_i^j(q) dq. \quad (7)$$

The calculation of failure probability $P_i^j(q)$ in Equation(7) concerns two stages: glitch generation and glitch propagation. $P_i^j(q)$ is estimated with the following Equation(8):

$$P_i^j(q) = F_{i\psi}(q) \cdot P_i^j(\psi), \quad (8)$$

where $F_{i\psi}(q)$ is a flag function indicates whether a glitch ψ (with enough amplitude) is generated when node i collects charge q ; and $P_i^j(\psi)$ is the probability that glitch ψ at node i can successfully propagate in the fan-out paths of node i and eventually be latched by a storage element at output terminal j . $F_{i\psi}(q)$ is defined as the following function:

$$F_{i\psi}(q) = \begin{cases} 0, & V_{max}(\psi) \leq V_{th} \\ 1, & V_{max}(\psi) > V_{th} \end{cases} \quad (9)$$

The characteristics of generated glitch are obtained from circuit simulation by attaching current source to the struck node, as aforementioned in Figure 1. The current source used in this work is a single exponential form described in Equation(10) [32], where q is the collected charge and τ is a technology-dependent time constant.

$$I_q(t) = \pm \frac{2q}{\tau\sqrt{\pi}} \sqrt{\frac{t}{\tau}} e^{-\frac{t}{\tau}}. \quad (10)$$

Previous works usually use gate as the basic block in the analysis, e.g., in Equation(7) the A_i and $P_i^j(q)$ all refer to gate i , which propose some limitations on the analysis. [7] has reported charge collection slope Q_s for PMOS and NMOS are distinctive and dramatically affect the striking rate calculation. However, using gate as the basic block forces unique Q_s for PMOS and

NMOS striking rate calculation. Furthermore, the method appending current source at the gate output is also incomplete because the glitch generation depends on many factors: the transistor being struck, gate status, loads, etc. Figure 3 depicts a 130 nm BUFX8 cell being struck. When collected charge q is less than 80 fC, the largest transistors M0 and M1 are insensitive (glitch amplitude is lower than the threshold) to the strike, and only the smaller M3 with input A at “0” state is sensitive to particle hitting. If we use gate as the basic analysis block and model glitch generation as in Figure 1(a), then no glitches will be observed at the gate output, thus affecting the SER estimation. In this work, individual transistors were used as the basic analysis node, thus all the aforementioned factors affecting glitch generation and propagation were addressed.

Figure 3: Glitch generation dependence on transistor

Generated glitches will propagate in the DUT and can be affected by the three masking mechanisms, thus the probability $P_i^j(\psi)$ in Equation(8) is estimated with the following equation:

$$P_i^j(\psi) = P_\psi(LM \cap EM \cap LW), \quad (11)$$

where LM, EM and LW are three events: LM represents the glitch ψ has not been filtered out by logic masking (depending on circuit states) of fan-out paths; EM represents the glitch suffering electrical masking but still keeping

enough width and amplitude; LW indicates the glitch is latched by flip-flop at the output j and upsets the flip-flop eventually.

Have considered the striking granularity and gate state dependence problem, Equation(7) for combinational logic SER estimation is changed to the following form:

$$SER_{comb} = F \cdot k \cdot \sum_{i=1}^G \sum_{m=1}^{M_i} \sum_{j=1}^O \int_{s \in S_i} \int_{Q_{min}}^{Q_{max}} A_m \cdot \frac{1}{Q_{sm}} \cdot e^{-\frac{q}{Q_{sm}}} \cdot F_{i\psi}(q) \cdot P_i^j(\psi) dq ds, \quad (12)$$

where G is the maximum gate number; M_i is the transistor number of gate i , S_i is the state space of gate i ; Q_{sm} is the charge collection slop of transistor m depending on its type (PMOS/NMOS). The integration in Equation(12) might be hard to calculate, thus in our final implementation, the integration was converted to summation operations.

3.3. Masking Probability Calculation

Equation(11) is determined by logical masking, electrical masking and latching-window masking, which can be resolved by the statistical approach. In ASSET-VLG, $P_i^j(\psi)$ in Equation(11) is estimated by two approaches: vector mode and probability mode. The main difference between these two modes is the evaluation of logical masking factor. In vector mode, a set of input patterns randomly generated (or user specified) are fed into ASSET-VLG one by one to evaluate the SER. For each input pattern, the DUT circuit is traversed and glitch generating and propagating are performed for each gate sequentially, then $F_{i\psi}(q)$, $P_i^j(\psi)$ and $P_i^j(q)$ of Equation(8) for current pattern are obtained. When all the patterns have been propagated, the expectation of $P_i^j(q)$ is treated as the final result of Equation(8). However, given N input patterns, the analysis requires N passes of circuit traversal, which is time consuming and limits the performance of vector mode.

ASSET-VLG also supports a simpler mode — probability mode that the failure probability $P_i^j(q)$ is calculated by only 1-pass propagation. Probability mode assumes LM, EM and LW are independent events, thus Equation(11) is simplified to the following equation:

$$P_i^j(\psi) = P_\psi(LM) \cdot P_\psi(EM) \dot{P}_\psi(LW), \quad (13)$$

where $P_\psi(LM)$, $P_\psi(EM)$ and $P_\psi(LW)$ are three probabilities that the glitch survived after the propagation. In probability mode, $P_\psi(LM)$ is evaluated from the signal probability (the probability a node at “0/1” state) that can be obtained in advance. Probability mode reduces circuit traverse to only one pass and provides better speed than vector mode with limited accuracy because LM, EM and LW may not be independent.

In both modes, each glitch is associated with a property — the propagation probability $P(LM)$ to assess the logical masking factor. This probability will be updated when glitch passes through a gate according to the rules listed in Table 1, where F in vector mode is a flag function degenerated from probability function. $F(LM) = 0$ indicates the glitch is logical masked and $F(LM) = 1$ means the glitch is unaffected by logical masking under current input pattern. The superscript i means the glitch enters gate through input port i , and subscript I/O indicates the original and updated probability after propagation. INV, BUF, XOR and XNOR gates are immune to logical masking, but other gates are sensitive to logical masking. Considering a AND gate, if any of its input port other than i is at “0” state, the glitches entered port i will not pass through this gate and be totally filtered out by logical masking.

Electrical masking is investigated by a LUT storing input glitches and their corresponding output glitches for all gate types and loads. The LUT is constructed with spice simulator invoked by an automatic tool. For a glitch $\langle w_i, h_i \rangle$ propagated to the gate, corresponding output glitch $\langle w_o, h_o \rangle$ is queried from the LUT where w and h means the glitch width and height separately. If $w_o > 0$ and $h_o > V_{th}$, $P_\psi(EM)$ is regarded equal to 100%, otherwise $P_\psi(EM) = 0$ and the input glitch is evaporated.

The latching probability of glitch reaches DFF is evaluated by the following empirical formula:

$$P_\psi(LW) = \begin{cases} 0, & T_\psi < T_{lw} \\ \frac{(T_\psi - T_{lw})}{T_{clk}}, & T_\psi \geq T_{lw}, \end{cases} \quad (14)$$

where T_ψ is the width of glitch ψ reaching the DFF, T_{lw} is the latching-window (sum of setup time and hold time) width of DFF and T_{clk} is the clock width of the DUT.

3.4. Sequential Logic SER

[6] divided a classical master-slave DFF into setup stage and hold stage,

Table 1: Logical Masking Factor Evaluation

Gate Type	Vector Mode	Probability Mode
INV	$F_O(LM) = F_I(LM)$	$P_O(LM) = P_I(LM)$
BUF	$F_O(LM) = F_I(LM)$	$P_O(LM) = P_I(LM)$
AND	$F_O^i(LM) = F_I^i(LM) \cdot \prod_{\substack{j=1 \\ j \neq i}}^N F(X_j = 1)$	$P_O^i(LM) = P_I^i(LM) \cdot \prod_{\substack{j=1 \\ j \neq i}}^N P(X_j = 1)$
NAND	$F_O^i(LM) = F_I^i(LM) \cdot \prod_{\substack{j=1 \\ j \neq i}}^N F(X_j = 1)$	$P_O^i(LM) = P_I^i(LM) \cdot \prod_{\substack{j=1 \\ j \neq i}}^N P(X_j = 1)$
OR	$F_O^i(LM) = F_I^i(LM) \cdot \prod_{\substack{j=1 \\ j \neq i}}^N F(X_j = 0)$	$P_O^i(LM) = P_I^i(LM) \cdot \prod_{\substack{j=1 \\ j \neq i}}^N P(X_j = 0)$
NOR	$F_O^i(LM) = F_I^i(LM) \cdot \prod_{\substack{j=1 \\ j \neq i}}^N F(X_j = 0)$	$P_O^i(LM) = P_I^i(LM) \cdot \prod_{\substack{j=1 \\ j \neq i}}^N P(X_j = 0)$
XOR	$F_O^i(LM) = F_I^i(LM)$	$P_O^i(LM) = P_I^i(LM)$
XNOR	$F_O^i(LM) = F_I^i(LM)$	$P_O^i(LM) = P_I^i(LM)$

and calculated the SER due to particle striking these two stages individually with various methods. On the contrary, here we calculate them with unique methods. Setup stage SER of a DFF is obtained with the following equation:

$$SER_{setup} = F \cdot k \cdot \sum_{i=1}^M \left(A_i \cdot \int_{Q_{min}}^{Q_{max}} \frac{1}{Q_{si}} \cdot e^{-\frac{q}{Q_{si}}} \cdot P_i(q) dq \right). \quad (15)$$

where M is maximum transistor number of the setup stage, and $P_i(q)$ is the probability that transistor i being struck and upset the DFF itself. $P_i(q)$ is affected by the hitting time and DFF status [33, 34]. In [6], hitting time dependence is modeled with uniform distribution but state dependence is not mentioned.

DFF hold stage SER calculation can be simplified to the following form utilizing the critical charge concept:

$$SER_{hold} = F \cdot k \cdot \sum_{i=1}^M \left(A_i \cdot \int_{Q_{crit}}^{Q_{max}} \frac{1}{Q_{si}} \cdot e^{-\frac{q}{Q_{si}}} dq \right), \quad (16)$$

where Q_{crit} is the critical charge when collected charge q exceeds Q_{crit} will definitely upset the DFF, which can be calculated via circuit simulation to determine the minimal charge changing the node state [35]. Usually DFF has larger Q_{crit} and is less sensitive to striking than SRAM cell [4]. Equation(16) also removes the $P_i(q)$ from Equation(15) because it assuming hold stage SER is insensitive to hitting time and DFF status.

The above approach may be appropriate for classical master-slave DFFs. But there are lots of other types of flip-flops in an industrial standard cell library, such as flip-flops with set/reset, scan structure or designed for high performance/low power, etc. These flip-flops might have transistors not covered by the above methods. Therefore we combined Equation(15) and (16) to obtain a unique equation to calculate the SER of whole DFF:

$$SER_{dff} = F \cdot k \cdot \sum_{i=1}^M \left(A_i \cdot \int_{Q_{min}}^{Q_{max}} \frac{1}{Q_{si}} \cdot e^{-\frac{q}{Q_{si}}} \cdot P_i(q) dq \right). \quad (17)$$

Equation(17) unifies the SER estimation for DFF setup stage and hold stage. The only difference for these two stages is the calculation of $P_i(q)$, which is addressed by statistical fault injection in later implementation sections.

4. ASSET-VLG Framework

4.1. Overview of ASSET-VLG

The structure of ASSET-VLG is depicted in Figure 4, where the work flow is divided into three steps: i) standard cell library modeling; ii) auxiliary files preparation; iii) SER analysis. Library modeling for a technology only needs execute once, including glitch generation characteristics modeling, glitch propagation characteristics modeling and DFF SER modeling. The details of library modeling are described in the following sections. Auxiliary files indicate the files needed for each DUT analysis. The auxiliary files include the random input patterns and the signal probability data.

For an analysis, verilog netlist of DUT should be obtained from logic synthesizer firstly. The netlist is parsed by a fully functional verilog parser developed by us with ANTLR tool [36]. The verilog parser uses syntax directed translation techniques to exploit the cells, ports, nets and their connections. Another important role of the parser is to partition the DUT to combinational part and sequential part, which is accomplished by extracting

Figure 4: Structure of ASSET-VLG

all DFFs out and converting D terminal of the DFFs to the output ports of combinational part, and convert Q or QN terminal of the DFFs to the input ports. The ports and nets data from verilog parser are also utilized to generate random input patterns. With the ports and nets data, plus user specified clock rate and patterns number, a Python script is employed to generate a verilog format test bed for the DUT. The test bed has codes to randomly generate input patterns for all input ports. The script will also generate shell scripts to call logic simulator and dump all the nets and output ports states to a pattern file. The pattern file is utilized to calculate the logical masking factor in vector mode analysis, or be processed further to generate signal state probabilities for probability mode SER analysis.

The DFFs extracted by parser are stored in a list for later sequential part analysis, while the combinational part are converted to graph representation form. Then topological sorting algorithm is carried out for the graph to obtain a partial ordered gate list for combinational SER analysis. The following step is to calculate the combinational SER by vector mode or probability mode. These two modes analyses are described in the bottom-right dash line boxes of Figure 4. The main difference between vector mode and probability mode is vector mode needs multiple input patterns to be propagated in the topological sorted circuit graph.

The detailed procedure of vector mode SER analysis is depicted in Algorithm 1. For each gate under current input pattern, all the transistors in this gate was traversed to investigate the glitch generation when charge q is collected. The width and height of the glitch generated are queried from the glitch generation LUT LUT_G . In addition to the glitches generated, there are also glitches propagated from fan-in gates to the current gate. $Glit_{fanin}$ are the glitches propagated from fan-in gates and be logically masked and electrically masked by the current gate. Logical masking is preformed according to Table 1, while electrical masking is investigated by querying the glitch propagation table LUT_P to find out corresponding glitch width and height after electrical masking. Then the glitches generated, along with masked fan-in glitches, are propagated to fan-out gates. When all gates have been processed, a new input pattern is read in and repeat the glitch generation and propagation again. When all input patterns have been propagated, glitches residing in the output ports are traversed to calculate the latching probability and the combinational SER.

The analysis for probability mode is simpler than vector mode because the glitch generation and propagation are performed for each gate only once. The

Algorithm 1: Vector mode SER analysis for combinational part

Input: DUT's graph G , input patterns V , glitch generation table LUT_G and glitch propagation table LUT_P .

Output: Soft error rate SER_{comb} .

```
for  $i \leftarrow 1$  to  $|V|$  do                                     /* For each pattern */
  for  $j \leftarrow 1$  to  $|G|$  do                                 /* For each gate */
     $Glit_{gen} = \emptyset$ ;
    for  $k \leftarrow 1$  to  $|G_j|$  do /* For each transistor in gate */
      for ( $q \leftarrow Q_{min}, q \leq Q_{max}, q+ = \delta q$ ) do /* For each
        charge */
           $Glit_{gen} += \text{QueryGlitGen}(G_j, M_k, q, V_i, load)$ ;
        end
      end
       $Glit_{lm} = \text{LogicalMask}(G_j, V_i, Glit_{fanin})$ ;
       $Glit_{em} = \text{ElectricalMask}(G_j, V_i, load, Glit_{lm})$ ;
       $\text{ForwardProp}(Glit_{gen}, Glit_{em})$ ;
    end
  end
   $SER_{comb} = 0$ ;
  for  $u \leftarrow 1$  to  $|O|$  do /* For each primary output */
    foreach  $\psi$  in  $O_u$  do /*  $\psi$  is a glitch */
       $SER_{comb} += \text{CalcLatchProb}(\psi)$ ;
    end
  end
   $SER_{comb} /= |V|$ ;
```

logical masking is calculated based on the signal probability. The procedure of probability mode is depicted in Algorithm 2. For each gate, all its possible input state space should be exhaustively enumerated and investigated, e.g., for 2-input gate four states should be considered: “00”, “01”, “10”, “11”. The glitch generation and glitch propagation should also consider all these states combination.

4.2. Lookup Table Construction

One obvious drawback of Equation(10) is it can't be modeled by spice directly with exponential current source. ASSET-VLG use piece-wise linear

Algorithm 2: Probability mode SER analysis for combinational part

Input: DUT's graph G , input patterns V , glitch generation table LUT_G and glitch propagation table LUT_P .

Output: Soft error rate SER_{comb} .

```
for  $j \leftarrow 1$  to  $|G|$  do                                     /* For each gate */
   $Glit_{gen} = \emptyset$ ;
   $S = \text{GenStateSpace}(G_j)$ ;
  for  $k \leftarrow 1$  to  $|G_j|$  do                               /* For each transistor in gate */
    for  $(q \leftarrow Q_{min}, q \leq Q_{max}, q += \delta q)$  do /* For each charge */
      /*
       $Glit_{gen} += \text{QueryGlitGen}(G_j, M_k, q, S, load)$ ;
      */
    end
  end
   $Glit_{lm} = \text{LogicalMask}(G_j, S, Glit_{fanin})$ ;
   $Glit_{em} = \text{ElectricalMask}(G_j, S, load, Glit_{lm})$ ;
   $\text{ForwardProp}(Glit_{gen}, Glit_{em})$ ;
end
 $SER_{comb} = 0$ ;
for  $u \leftarrow 1$  to  $|O|$  do                                  /* For each primary output */
  foreach  $\psi$  in  $O_u$  do                                     /*  $\psi$  is a glitch */
     $SER_{comb} += \text{CalcLatchProb}(\psi)$ ;
  end
end
end
```

approximation to model it instead. ASSET-VLG extensively uses LUT in the analysis for fast querying glitches being generated or being attenuated in propagation. In order to improve the analysis performance, hash table are utilized as the basic data structure to implement the lookup tables. The hash table for glitch generation and glitch propagation are depicted in Figure 5 separately.

The hash tables both use a 64 bit integer as the hash key. The width and height of glitches generated by particle striking are affected by cell type, collected charge, gate load, gate state and struck transistor, hence we use cell type, charge, load and state to composite the hash key by quantifying and shifting operations on these factors. The hash value for glitch generation is a list of glitches for each transistor being struck. The hash key for glitch

(a) LUT for glitch generation

(b) LUT for glitch propagation

Figure 5: Hash table structures for LUT

propagation makes up from cell type, port, input width, input height and current load, while the hash value is the output glitch width and height after attenuation by the gate.

Two Python scripts were developed to automatically call the spice simulator to generate the hash items of the LUT. The number of spice simulation depends on multi-factors combination. In this work the cell type has about 170 candidates; charge is in the range [10 fC, 150 fC] [7] (9 to hundreds of fC in [37] and minimal 21 fC in [38] for 130 nm technology); loads falls into [1 fF, 200 fF]; gate state may has maximal 16 combinations for 4-input gates. Assuming each gate has 10 transistors on average, thus the combination of above factors is about $7.6 E8$, which is too large to finish the simulation in reasonable time. Hence we use a simplified sampling scheme to reduce the demanded simulations. For collected charge, only the values divisible by 10 are investigated; while step-varied sampling technique is employed on load, glitch width and height, e.g., 1 fF interval for load smaller than 20 fF and 10 fF interval for [20 fF, 200 fF] are adopted to reduce sample points. Combined with pruning techniques to filter out the zero output samples, $7.4 E5$ spice simulations for glitch generation LUT and $5 E6$ simulations for glitch propagation LUT are executed within about 1 week.

Because only some discrete points were investigated for glitch generation and propagation, thus interpolation techniques were utilized to predict

the glitch waveform for the points not existing in LUT. In glitch generation, simple linear interpolation is carried out for load, while trilinear interpolation [39] is used for width, height and loads in glitch propagation. As an example, consider a glitch (218 ps/1180 mv) passing through a gate AND2XL at A port with 3.5 fF load connected to the gate, where 8 items should be queried from the LUT to perform the trilinear interpolation. The 8 items are: (200 ps, 1100 mv, 3 fF), (200 ps, 1100 mv, 4 fF), (200 ps, 1200 mv, 3 fF), (200 ps, 1200 mv, 4 fF), (220 ps, 1100 mv, 3 fF), (220 ps, 1100 mv, 4 fF), (220 ps, 1200 mv, 3 fF), (220 ps, 1200 mv, 4 fF). Similar technique is also used in EDA field.

In order to verify the accuracy of the interpolation techniques, approximately 300 random samples are simulated by spice and predicted by interpolation separately for glitch generation and glitch propagation. Figure 6 gives the comparison between spice simulation and interpolation results, where lines are spice results, dots and squares are interpolated results. Interpolated results match with spice simulation results well with 98.4% accuracy for glitch generation and 98.0% accuracy for glitch propagation. There are still some points seriously deviated from simulation results by 10%. However, all these points are for large load exceeding 100 fF, which rarely appears in practical logic circuits in 130 nm technology. Thus the interpolation techniques provide at least 98% accuracy compared to spice even if the bad points are took into account.

Figure 6: Comparison between spice and interpolation results

4.3. Glitch Pruning

SER estimation technique depending on glitch propagation has a disadvantage that there are too many glitches propagating in the circuit graph representation, especially when the DUT has complex branch and reconvergent paths. In probability mode of ASSET-VLG, the glitches for all state combinations should be stored in the circuit graph data structure, which makes the glitches exceed 1 million in one gate and brings critical challenge for memory capacity. The most obvious case exists in ISCAS'85 C6288 circuit. In our machine with 8 GB memory, it will run out of all the physical memory.

In order to circumvent this problem, a grid based technique is employed to prune the glitches. Considering a surface of glitch width and height, we divide the surface into a grid containing about 5000 lattices, where the width axis is divided into 10 ps divisions and height axis is divided into 100 mv divisions. For glitches fall into same lattice, the lattice central point is used to represent all the glitches of this lattice.

Figure 7(a) and (b) use scatter graphs to illustrate the number of glitches propagated in C6288, where the first subgraph is for the original version and the second subgraph is for the optimized version. In the original version, the program run out of memory after propagating half of the internal nodes, at that time many nodes have stored millions of glitches. The optimized result is depicted in Figure 7(b), where the glitches in each gate have been pruned to about 5000 and reduce the memory consumption significantly by about 3 order magnitudes.

Figure 7: Glitch pruning for C6288

4.4. DFF SER Estimation

As discussed in previous section, the DFF SER depends on clock rate and input pattern, thus we combine statistical fault injection and curve fitting techniques to rapidly estimate the given DFF's SER. The statistical fault injection was carried out based on a Python program developed by us, which repeatedly call spice simulator to simulate the DFF with randomly generated input pattern for various hitting time, **transistor**, collected charge and clock rate combinations. For each DFF cell in the library, a randomly generated input pattern is created and the "0/1" ratio is about 50%. To be simple, for our 130 nm technology, 8 clock rates are investigated. These clock frequencies spread in the range 100 MHz–1000 MHz, covering the commonly used frequency range. For each frequency, the SER of DFF is estimated by statistical fault injection according Equation(17).

After obtaining the SER of each DFF at 8 clock rates, the SER for clock rate other than pre-investigated frequencies is predicated by linear regression. After linear regression, we obtain a pair of $< k, b >$ where k is the slope and b is the interception. For random clock rate x in MHz, the SER at this frequency is obtained by $kx + b$. Figure 8(a) and (b) show two types of DFF's SER and the regression result, where each DFF has 4 distinctive sizes and 4 regression results. The figure shows better matching of the regression approach.

Figure 8: Two Curve fitting Samples for DFF SER

In [12], the author reported the frequency dependence of alpha particle

Table 2: DFF SER (Fit/bit) Comparison for 130 nm Technology

[13]	[40]	[41]	This work
$1E - 5 \sim 1E - 4$	$\approx 2E - 3$	$\approx 5E - 4$	$0.6E - 4 \sim 2.8E - 4$

induced SER of DFF in 40 nm technology, where the slopes of DFF SER is about 0.14 FIT/(Mbit×MHz). As a comparison, the slopes of Figure 8 are between 0.11 to 0.18 FIT/(Mbit×MHz), which are similar to [12]. In order to verify our approach further, Table 2 lists classical DFF SER predicted for 130 nm, where data of [13, 40] is derived from the reference directly, but the result of [41] is extrapolated from 40 nm according to the technology trends of [40]. From Table 2 we can see our results fall into the reasonable range reported by previous works.

4.5. Parallel of ASSET-VLG

Parallelization can accelerate the SER estimation and make the tool applicable to much larger circuits. However, the parallelizing of ASSET-VLG is mainly for vector mode rather than probability mode. The probability mode analysis is hard to accelerate because it is mainly a graph traverse algorithm with complex data structure (graph and hash table) and bigger working set. Complex data structure and bigger working set is inappropriate for GPGPU like fine-grained parallelization. On the other hand, there are not enough coarse-grained parallelism in probability mode analysis to cover the cost of thread context switching for multi-threading parallelization. Hence we only exploited the parallelism in vector mode analysis by utilizing the power of multi-core platforms.

Profiling analysis for vector mode program shows most of the cpu time is consumed at glitch propagation, especially the glitch querying from LUT. Hence there are three places to accelerate the program execution: i) trilinear interpolation function with 8 hash table query operations; ii) the electrical masking code where the trilinear interpolation operation need be invoked for every glitch; iii) the first outermost for loop in Algorithm 1. For the trilinear interpolation, we can assign each LUT querying an individual thread. In electrical masking each glitch can be processed by an individual thread also. Unfortunately, there are no sufficient calculations in these two conditions to amortize the thread context switching cost. We have tried to parallelize the program for these two conditions by OpenMP and boost library thread, but both of them show nearly 50% performance degradation. Therefore multi-

threading parallelization is carried out for the third case, where each input pattern is totally processed by an individual thread. The processing includes glitch generation, glitch propagation and latching probability calculation, and dramatic speedup is obtained in the experiments.

5. Experiments and Results

5.1. Experiments Setup

All the development works were finished in Windows environment, although most of the components have been tested in Redhat RHEL 6 platform. The kernel analyzer of ASSET-VLG is developed with C++ and AntLR, thus can be compiled and executed in Windows or Linux platform. Other components of ASSET-VLG are developed with Python and can be executed in either Windows or Linux platform without any modification.

The target technology in this work is a industrial 130 nm library, and the cell basic parameters are collected by hand from the library. The glitch generation and propagation LUT were constructed for ordinary INV, BUF, AND/NAND, OR/NOR, XOR/XNOR gates with the help of our Python based automatic tools. The library modeling tools invoke spice simulator and measure the glitch automatically to liberate us from cumbersome handwork. The finished glitch generation LUT has less than one million items and there are 5 millions items in the glitch propagation LUT. The DFFs modeling was also performed with automatically statistical tool, and each DFF demands at least 10 thousands spice simulations, then curve fitting program written with Python was invoked to get the slope and interception for SER predicting. The library modeling consumed about one week, but these works only need execute once for each technology.

The benchmark circuits selected were ISCAS'85 combinational circuits and ISCAS'89 sequential circuits. The circuits were synthesized to gate level netlists by Synopsys Design Compiler for the 130 nm target library. The input patterns file and signal probability file were generated for each benchmark circuit by provided automatic tools and commercial logic simulator. For each circuit, 10000 random input patterns were generated and the signal probability file was extracted from the input pattern file. The parameters for the SER analysis are listed as follows: neutron flux is set to sea level $56.5 m^{-2}s^{-1}$; the effective particle hit rate k of previous equations is set to 2.2×10^{-5} [31]; charge collection slope for PMOS and NMOS are set to 6.5 fC and 17.3 fC separately [7]; the clock frequency for all circuits are set

Table 3: SER Results of ISCAS’85 Benchmark Circuits

Circuit	#Gate	#Input	#Nets	Area (μm^2)	SER _V (FITs)	SER _P (FITs)	Time _V (s)	Time _P (s)
C17	6	5	11	3.23E+01	7.51E-06	7.25E-06	8.23E-02	5.09E-04
C432	212	36	248	1.79E+03	6.02E-05	5.77E-05	6.24E+00	8.13E-02
C499	248	41	289	3.00E+03	1.33E-03	1.35E-03	2.71E+01	1.08E-01
C880	303	60	363	2.28E+03	3.41E-04	3.48E-04	1.41E+01	2.68E-01
C1355	287	41	328	3.07E+03	1.24E-03	1.29E-03	2.77E+01	1.23E-01
C1908	287	33	320	3.24E+03	5.26E-04	5.84E-04	1.69E+01	2.35E-01
C2670	550	233	784	4.20E+03	5.86E-04	5.94E-04	1.74E+01	3.04E-01
C3540	865	50	915	7.06E+03	9.09E-04	8.91E-04	4.91E+01	3.63E-01
C5315	1136	178	1314	8.38E+03	2.27E-03	2.29E-03	6.85E+01	4.73E-01
C6288	3181	32	3216	2.95E+04	3.07E-03	2.71E-03	5.55E+02	2.16E+00
C7552	1652	207	1859	1.62E+04	1.36E-03	1.31E-03	1.07E+02	1.03E+00

to 500 MHz uniformly; for the non-registered output we assume a virtual basic DFFX1 is attached and its latching-window is about 110 ps. Other parameters such as transistor area and input capacitance were extracted from technology document and stored in our library.

The hardware platform for the analysis is a PC workstation equipped with one Intel Xeon E3 (4 cores 8 simultaneous threads) processor and 8 GB memory. The final SER analysis is carried out in windows environment, and the high accuracy performance measure is performed with boost timer library. All the circuits are analyzed for vector mode and probability mode, while in vector mode 1, 2, 4, 8 threads have been investigated to calculate the speedup.

5.2. Results and Analysis

5.2.1. SER Results

The SER results for ISCAS’85 and ISCAS’89 benchmark circuits are listed in Table 3 and Table 4, where the “SER_V”, “SER_P”, “Time_V”, Time_P are SER and optimized run time for vector mode and probability mode separately. From Table 3 and Table 4, ASSET-VLG can analyze the SER of gate level verilog netlist in acceptable time.

In Table 3, the variation of SER between vector mode and probability mode are 1.13% $\sim$ 11.62%. The analysis speed of probability mode is faster than vector mode by at least one order of magnitude. However, the

Table 4: SER Results of ISCAS'89 Benchmark Circuits

Circuit	#Gate	#FF	#Nets	Area (μm^2)	SER _V (FITs)	SER _P (FITs)	Time _V (s)	Time _P (s)
S27	7	3	15	1.15E+02	3.80E-04	3.81E-04	1.14E-01	6.91E-04
S298	73	14	94	8.16E+02	1.76E-03	1.75E-03	1.26E+00	8.58E-03
S444	94	21	120	1.10E+03	2.70E-03	2.71E-03	2.46E+00	1.17E-02
S526	121	21	148	1.29E+03	2.85E-03	2.87E-03	3.05E+00	1.40E-02
S1196	384	18	417	2.74E+03	2.38E-03	2.42E-03	1.35E+01	1.15E-01
S1238	383	18	416	2.76E+03	2.32E-03	2.40E-03	1.09E+01	7.71E-02
S13207	1432	625	2140	2.83E+04	7.75E-02	7.75E-02	3.33E+01	1.12E-01
S38584	7814	1275	9221	8.19E+04	1.67E-01	1.67E-01	3.40E+02	1.90E+00

probability mode analysis assume logical masking is independent from other masking events, which may introduce error into the SER estimation. Furthermore, probability mode has not considered the system SER under practical workloads and limit its applicability, only if the analysis speed is the most important problem.

Comparing Table 4 with Table 3, the SER of ISCAS'89 circuits are much higher than ISCAS'85 circuits with similar area. This abnormal phenomenon indicates that sequential logic in 130nm technology is more sensitive than combinational logic and dominates the system SER. Figure 9 shows the SER comparison between the sequential part and combinational part of the same circuit, where SER of sequential logic are about one order higher in magnitude. Same trends were also observed in our another research that analyzed the SER of ISCAS benchmark circuits by using circuit level SER evaluation tool ASSET-SPI on a 180nm commercial library. As indicated in [15], sequential logic soft errors still dominate at features sizes as small as 32nm. This work reveals the sequential logic domination on soft error by analytical method rather than expensive bombarding approaches, reminding us the hardening of sequential logic is the critical issue in soft error tolerant design.

5.2.2. Glitch Pruning Results

Glitch pruning was also carried out and its effects on SER value, memory usage and analysis time were also investigated. In vector mode, all of the circuits can be analyzed successfully, and glitch pruning has no influence on the estimated SER value. The analysis time and memory usage also make

Figure 9: Sequential logic SER vs. combinational logic SER

no change after pruning for all circuits except C6288. C6288 is a multiplier with complex branch and reconvergent paths, in which the memory usage is reduced to 15% of non-pruned running and analysis time is slightly reduced to 79%. The results indicate that the glitch amount is not a critical issue at most SER analyses in vector mode.

In probability mode, the SER estimation error induced by glitch pruning is 0.34% only, but its effects on memory usage and analysis time are dramatic. Because ASSET-VLG uses quite large glitch generation and glitch propagation LUT, even the smallest C17 circuits with only 6 gates still require 258 MB memory to store the LUT and some other auxiliary data structures, thus in the following investigation we subtract 258 MB from whole memory usage to obtain the glitch pruning affection. Figure 10 depicts the glitch pruning results, where in each group the first two bars show the memory usage optimization effects and the last two bars show the analysis run time reduction. Note C6288 only gives pruned results because in non-pruned analysis the program is forced to terminate for memory explosion. Figure 10 shows glitch pruning can dramatically reduce the memory usage and time consuming with only negligible 0.34% accuracy lost in probability mode.

5.2.3. Speedup of Parallelization

The speedup of vector mode analysis for 2, 4 and 8 parallel threads are illustrated in Figure 11, where the three bars in one group are speedup of 2, 4, 8 threads separately. The most interesting thing in Figure 11 is the superlinear speedup in C17, which can be attributed to the cache locality for a later iteration fetching the glitches directly from processor cache rather than the slow main memory. The superlinear speedup disappears in the other

Figure 10: Glitch pruning affections on memory usage and analysis time

larger circuits. Excluding C17 with superlinear speed, the average speedup for 2, 4 and 8 threads are 1.89, 3.41, 4.44 separately. Because the processor has only 4 cores and 2 simultaneous threads in each physical core, and there is only one suite of memory access infrastructure for each core, hence the speedup for 8 threads is only 4.44. If we want higher speedup, partitioning the input patterns to multiple workstations to exploit the power of cluster computing is a feasible approach.

Figure 11: Speedup in vector mode

6. Conclusions

Soft error dominates modern highly reliable computing systems and demands fast, accurate and scalable SER estimation methodologies in the early design stage. This work proposed a systematic and automatic logic level SER estimation tool ASSET-VLG. ASSET-VLG automates the analysis flow by

compiler techniques and provides two flexible working modes: vector mode and probability mode. The vector mode of ASSET-VLG support accurate analysis and practical input patterns, while probability mode provides similarly results with at least one order of magnitude faster analysis speed.

Unlike previous works targeted at predictive model libraries and focused on combinational logic circuits, ASSET-VLG targets at industrial standard cell library and preforms automatic SER analysis for both combinational logic and sequential logic within a uniform SER estimation framework. As a framework focusing on practical application, ASSET-VLG completely considers the factors affecting glitch generation and propagation including transistor, gate state, load, etc. The fast SER estimation of sequential logic is addressed by pre-characterizing and curve fitting on classical DFF cells. The characterizing SER results of DFF by our analytical approach matches well with previous expensive physical SER evaluation methodologies. Experiments on ISCAS'85 and ISCAS'89 benchmark circuits reveal the feasibility of our approach. The analyses on ISCAS'89 benchmark circuits show sequential logic still dominates the whole circuit SER, which agrees with the previous predictions and reminds sequential logic should be hardened as the critical component rather than combinational logic.

Furthermore, glitch pruning technique to resolve memory explosion problem, and multi-threading parallelization for scalability are also exploited to enhance the applicability. Experiments show ASSET-VLG can analyze moderate gate level logic circuits in seconds for probability mode and minutes for vector mode, even for the larger C6288 or S38584 with thousands of cells. The glitch pruning technique can reduce memory usage and analysis time by one order with only 0.34% accuracy lost, and addresses the memory explosion problem in C6288 like circuits. The parallelizing efforts for vector mode by multi-threading techniques implies the methodology in this work has better scalability. When parallelized threads number are 2, 4 and 8, average 1.89, 3.41 and 4.44 speedup are obtained. Other than coarse-grained multi-threading parallelization, SER analysis may also be accelerated by fine-grained GPGPU techniques, but the memory intensive table query operations should be changed to computing intensive operations, which is our further research plan.

- [1] Y.-P. Fang, A. S. Oates, Muon-induced soft errors in SRAM circuits in the terrestrial environment, *IEEE Trans. Device Mater. Rel.* 15 (1) (2015) 115–122. doi:10.1109/TDMR.2015.2396673.

- [2] S. Mitra, T. Karnik, N. Seifert, M. Zhang, Logic soft errors in sub-65nm technologies design and CAD challenges, in: Proceedings of the 42nd Annual Conference on Design Automation (DAC'05), 2005, pp. 2–4.
- [3] H. T. Nguyen, Y. Yagil, N. Seifert, M. Reitsma, Chip-level soft error estimation method, *IEEE Trans. Device Mater. Rel.* 5 (3) (2005) 365–381.
- [4] R. C. Baumann, Radiation-induced soft errors in advanced semiconductor technologies, *IEEE Trans. Device Mater. Rel.* 5 (3) (2005) 305–316.
- [5] J. R. Schwank, M. R. Shaneyfelt, P. E. Dodd, Radiation hardness assurance testing of microelectronic devices and integrated circuits: Test guideline for proton and heavy ion single-event effects, *IEEE Trans. Nucl. Sci.* 60 (3) (2013) 2101–2118. doi:10.1109/TNS.2013.2261317.
- [6] M. Zhang, N. R. Shanbhag, Soft-error-rate-analysis (SERA) methodology, *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.* 25 (10) (2006) 2140–2155.
- [7] R. R. Rao, K. Chopra, D. T. Blaauw, D. M. Sylvester, Computing the soft error rate of a combinational logic circuit using parameterized descriptors, *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.* 26 (3) (2007) 468–479.
- [8] N. Miskov-Zivanov, D. Marculescu, Circuit reliability analysis using symbolic techniques, *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.* 25 (12) (2006) 2638–2649.
- [9] S. Krishnaswamy, S. M. Plaza, I. L. Markov, J. P. Hayes, Signature-based SER analysis and design of logic circuits, *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.* 28 (1) (2009) 74–86.
- [10] F. Wang, Y. Xie, Soft error rate analysis for combinational logic using an accurate electrical masking model, *IEEE Trans. Dependable Secure. Comput.* 8 (1) (2011) 137–146.
- [11] A. C.-C. Chang, R. H.-M. Huang, C. H.-P. Wen, Casser: A closed-form analysis framework for statistical soft error rate, *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* 21 (10) (2013) 1837–1848. doi:10.1109/TVLSI.2012.2220386.

- [12] S. Jagannathan, T. D. Loveless, B. L. Bhuva, N. J. Gaspard, N. Mahatme, T. Assis, S.-J. Wen, R. Wong, L. W. Massengill, Frequency dependence of alpha-particle induced soft error rates of flip-flops in 40-nm CMOS technology, *IEEE Trans. Nucl. Sci.* 59 (6) (2012) 2796–2802. doi:10.1109/TNS.2012.2223827.
- [13] 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, in: *Proceedings of the 2002 International Conference on Dependable Systems and Networks (DSN'02)*, 2002, pp. 389–398.
- [14] B. Gill, N. Seifert, V. Zia, Comparison of alpha-particle and neutron-induced combinational and sequential logic error rates at the 32nm technology node, in: *2009 IEEE International Reliability Physics Symposium*, 2009, pp. 199–205.
- [15] C. Slayman, Soft error trends and mitigation techniques in memory devices, in: *2011 Proceedings of Annual Reliability and Maintainability Symposium (RAMS'11)*, 2011, pp. 1–5.
- [16] H. H. K. Tang, SEMM-2: A new generation of single-event-effect modeling tools, *IBM J. RES. & DEV.* 52 (3) (2008) 233–244.
- [17] P. E. Dodd, Physics-based simulation of single-event effects, *IEEE Trans. Device Mater. Rel.* 5 (3) (2005) 343–357.
- [18] J.-C. Baraza, J. Gracia, S. Blanc, D. Gil, P.-J. Gil, Enhancement of fault injection techniques based on the modification of VHDL code, *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* 16 (6) (2008) 693–706.
- [19] H. Cho, S. Mirkhani, C.-Y. Cher, J. A. Abraham, S. Mitra, Quantitative evaluation of soft error injection techniques for robust system design, in: *2013 50th ACM / EDAC / IEEE Design Automation Conference (DAC'13)*, 2013, pp. 1–10.
- [20] A. KleinOsowski, E. H. Cannon, P. Oldiges, L. Wissel, Circuit design and modeling for soft errors, *IBM J. RES. & DEV.* 52 (3) (2008) 255–263.

- [21] Y. S. Dhillon, A. U. Diril, A. Chatterjee, A. D. Singh, Analysis and optimization of nanometer CMOS circuits for soft-error tolerance, *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* 14 (5) (2006) 514–524.
- [22] R. Rajaramant, J. S. Kim, N. Vijaykrishnan, Y. Xie, M. J. Irwin, SEAT-LA: A soft error analysis tool for combinational logic, in: 19th International Conference on VLSI Design (VLSID’06), 2006, p. 4 pp.
- [23] B. Zhang, W.-S. Wang, M. Orshansky, FASER: Fast analysis of soft error susceptibility for cell-based designs, in: Proceedings of the 7th International Symposium on Quality Electronic Design (ISQED’06), 2006, pp. 755–760.
- [24] S. Krishnaswamy, G. F. Viamontes, I. L. Markov, J. P. Hayes, Probabilistic transfer matrices in symbolic reliability analysis of logic circuits, *ACM Trans. Des. Autom. Electron. Syst.* 13 (1) (2008) 8–42. doi:10.1145/1297666.1297674.
- [25] H.-K. Peng, H.-M. Huang, Y.-H. Kuo, C. H.-P. Wen, Statistical soft error rate (SSER) analysis for scaled CMOS designs, *ACM Trans. Des. Autom. Electron. Syst.* 17 (1) (2012) 1–24. doi:10.1145/2071356.2071365.
- [26] H. Asadi, M. B. Tahoori, Soft error modeling and remediation techniques in ASIC designs, *Microelectronics Journal* 41 (8) (2010) 506–522. doi:http://dx.doi.org/10.1016/j.mejo.2010.06.002.
- [27] F. Wang, V. D. Agrawal, Soft error rate determination for nanoscale sequential logic, in: 2010 11th International Symposium on Quality Electronic Design (ISQED’10), 2010, pp. 225–230.
- [28] N. Miskov-Zivanov, D. Marculescu, Modeling and optimization for soft-error reliability of sequential circuits, *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.* 27 (5) (2008) 803–816.
- [29] D. Holcomb, W. Li, S. A. Seshia, Design as you see FIT: System-level soft error analysis of sequential circuits, in: Design, Automation & Test in Europe Conference & Exhibition (DATE’09), 2009, pp. 785–790.
- [30] S. S. Mukherjee, C. Weaver, J. Emer, S. K. Reinhardt, T. Austin, A systematic methodology to compute the architectural vulnerability factors for a high-performance microprocessor, in: Proceedings of 36th An-

- nual IEEE/ACM International Symposium on Microarchitecture (MICRO'03), 2013, pp. 29–40.
- [31] P. Hazucha, C. Svensson, Impact of CMOS technology scaling on the atmospheric neutron soft error rate, *IEEE Trans. Nucl. Sci.* 47 (6) (2000) 2586–2594. doi:10.1109/23.903813.
 - [32] L. B. Freeman, Critical charge calculations for a bipolar SRAM array, *IBM J. RES. & DEV.* 40 (1) (1996) 119–129.
 - [33] T. Heijmen, P. Roche, G. Gasiot, K. R. Forbes, D. Giot, A comprehensive study on the soft-error rate of flip-flops from 90-nm production libraries, *IEEE Trans. Device Mater. Rel.* 7 (1) (2007) 84–96. doi:10.1109/TDMR.2007.897517.
 - [34] D. Alexandrescu, E. Costenaro, A. Evans, State-aware single event analysis for sequential logic, in: 2013 IEEE 19th International On-Line Testing Symposium (IOLTS'13), 2013, pp. 151–156. doi:10.1109/IOLTS.2013.6604067.
 - [35] T. Heijmen, B. Kruseman, R. v. Veen, M. Meijer, Technology scaling of critical charges in storage circuits based on cross-coupled inverter-pairs, in: *Proceedings of 42nd Annual IEEE International Reliability Physics Symposium (IRPS'04)*, 2004, pp. 675–676.
 - [36] T. J. Parr, *The Definitive ANTLR Reference: Building Domain-Specific Languages*, Pragmatic Bookshelf, 2007.
 - [37] O. A. Amusan, L. W. Massengill, M. P. Baze, A. L. Sternberg, A. F. Witulski, B. L. Bhuvu, J. D. Black, Single event upsets in deep-submicrometer technologies due to charge sharing, *IEEE Trans. Device Mater. Rel.* 8 (3) (2008) 582–589. doi:10.1109/TDMR.2008.2000892.
 - [38] D. Rossi, J. M. Cazeaux, M. Omana, C. Metra, A. Chatterjee, Accurate linear model for SET critical charge estimation, *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* 17 (8) (2009) 1161–1166. doi:10.1109/TVLSI.2009.2020391.
 - [39] H. R. Kang, *Three-Dimensional Lookup Table with Interpolation*, SPIE Press Book, 2006, pp. 155–157.
URL <http://spie.org/samples/PM159.pdf>

- [40] A. Dixit, A. Wood, The impact of new technology on soft error rates, in: 2011 IEEE International Reliability Physics Symposium (IRPS'11), 2011, pp. 5B.4.1–5B.4.7.
- [41] D. Rennie, D. Li, M. Sachdev, B. L. Bhuva, S. Jagannathan, ShiJieWen, R. Wong, Performance, metastability, and soft-error robustness trade-offs for flip-flops in 40 nm CMOS, IEEE Trans. Circuits Syst. I, Reg. Papers 59 (8) (2012) 1626–1634. doi:10.1109/TCSI.2012.2206505.