SEAT-LA: A Soft Error Analysis tool for Combinational Logic

R. Rajaraman, J. S. Kim, N. Vijaykrishnan, Y. Xie, M. J. Irwin
Microsystems Design Laboratory, Penn State University
(ramanara, jskim, vijay, yuanxie, mji)@cse.psu.edu

Abstract

Radiation induced soft errors in combinational logic is expected to become as important as directly induced errors on state elements. Consequently, it has become important to develop techniques to quickly and accurately predict soft error rates (SER) in logic circuits. In this paper, we propose a new approach, which can be applied to designs that use cell libraries characterized for soft error analysis and utilizes analytical equations to model the propagation of a voltage pulse to the input of a state element. The average error of the SER estimates using our approach compared to the estimates obtained using circuit level simulations is 6.5% while providing an average speed up of 15000. We have demonstrated the scalability of our approach using designs from the ISCAS-85 benchmarks.

©ACM, 20XX. This is the author's version of the work. It is posted here by permission of ACM for your personal use. Not for redistribution. The definitive version was published in Proceedings of the 19th International conference on VLSI Design, *January 3-7, 2006* http://doi.acm.org/10.1145/nnnnnn.nnnnnn

©20xx IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

This work is supported in part by grants from GSRC and NSF (Career No. 093085)

SEAT-LA: A Soft Error Analysis tool for Combinational Logic[1]

R. Rajaraman, J. S. Kim, N. Vijaykrishnan, Y. Xie, M. J. Irwin
Microsystems Design Laboratory, Penn State University
(ramanara, jskim, vijay, yuanxie, mji)@cse.psu.edu

Abstract

Radiation induced soft errors in combinational logic is expected to become as important as directly induced errors on state elements. Consequently, it has become important to develop techniques to quickly and accurately predict soft error rates (SER) in logic circuits. In this paper, we propose a new approach, which can be applied to designs that use cell libraries characterized for soft error analysis and utilizes analytical equations to model the propagation of a voltage pulse to the input of a state element. The average error of the SER estimates using our approach compared to the estimates obtained using circuit level simulations is 6.5% while providing an average speed up of 15000. We have demonstrated the scalability of our approach using designs from the ISCAS-85 benchmarks.

1. Introduction

Soft errors are transient errors caused mainly due to high energy particle strikes from cosmic radiation. Such radiation directly or indirectly induces localized ionization capable of upsetting internal data states. The particle strikes can directly occur on state elements such as memories, flip-flops and latches and change their state. Additionally, state elements can latch incorrect values propagated from strikes that occur in combinational elements. With reducing pipeline depth and downscaling of nodal capacitance and supply voltages, radiation induced soft errors in combinational logic is gaining increasing attention and is expected to become as important as directly induced errors on state elements [1].

Consequently, it has become important to develop techniques to quickly and accurately predict SER in combinational circuits. Recently, there have been various approaches to estimate SER in logic circuits efficiently [1 - 4, 8]. In this paper, we propose a new approach to estimate SER for logic circuits that attempts to capture the three masking effects concurrently. A tool known as SEAT-LA (Soft Error Analysis Tool – Logic Analyzer), has been developed using the above methodology. This tool is a part of hierarchical Soft Error Analysis Toolset (SEAT) that supports different levels of abstractions.

The rest of the paper is organized as follows. In Section 2 a basic introduction to soft errors in logic circuits and related works is presented. Section 3 describes the methodology used in SEAT-LA for soft error analysis. The tool implementation is discussed in Section 4. Section 5 presents experimental verification of the tool and soft error estimates for some ISCAS circuits. Section 6 concludes the paper.

2. Soft Errors in Logic Circuits

High energy particles striking the silicon substrate generates electron hole pairs as they pass through the p-n junctions. These electron hole pairs generate short duration current pulses that cause soft errors. In memory circuits and latches, these errors flip the stored values while they cause transient glitches at the output of combinational circuits. In combinational circuits, errors occur when these transient glitches are latched by state elements [6].

In logic circuits, there are three inherent masking mechanisms that prevent the propagation of any given pulse along a path towards the input of a state element. *Logical, electrical* and l*atching window or time window masking* are the three masking effects that needs to be modeled for SER estimation in logic circuits. Recent works have proposed different methods to model the above masking effects [1-3, 8]. In [1] electrical masking is categorized into two further effects – increase in rise and fall time and delay degradation. It models these two separately and then combines them to model electrical masking. They however do not verify their methodology using device or circuit level analysis results. The SERA tool [2] combines probability theory, circuit simulation, graph theory and fault simulation to estimate SER. This tool characterizes inverter chains and extends the same to all gates which can result in inaccuracies. In contrast our approach characterizes each cell in the library. In [3], a mathematical model based on set-up and hold time was used for timing window (*tw*) estimation while the electrical masking effect was determined using noise rejection curves on various gates. Instead of noise rejection, we actually model the transfer of a glitch across the combinational logic. The tool ASERTA used in [8] models soft errors by modeling electrical masking using mathematical equations for pulse propagation. However they do not consider the effect of pulse heights on the electrical masking as we do in our work. Also our approach to re-convergent nodes is very different from the methodologies presented above. Thus, our approach is unique from all the previous approaches that have been proposed so far [1-3, 8]. There have been various works on glitch modeling for power consumption [5, 6]. These models, however, do not calculate all glitch properties such as pulse amplitude and width required for modeling SER. A mathematical expression to model glitch amplitudes as it propagates through logic gates is proposed in [7]. In this work, we propose a mathematical expression for output amplitude based on geometrical calculation on approximated input pulse widths. We also use a mathematical expression for the output pulse-width based

[1] This work is supported in part by grants from GSRC and NSF (Career No. 093085)

on approximating the output voltage to a trapezoidal or a triangular pulse.

3. Characterization and Methodology

3.1. Logic Cell Characterization

Our methodology assumes that a soft error upset is modeled by the injection of a current pulse. The first part of our characterization involves capturing the current-voltage transfer characteristics for the logic cells in our library for different current pulses occurring at the input nodes. The output of this characteristic table provides the output voltage pulse parameters that include the pulse magnitude, width and the rise and fall times of the output voltage pulse. The characterization is performed for different input and output capacitances and current pulses. The device level version of SEAT was used to determine the dominant type of current pulses for which this characterization needs to be done. One disadvantage of this approach is that for a large library the number of tables will be large.

The second pre-characterization involves delay characterization for all the cells in our library. This characterization may already be available for the target cell library. Our analytical models for calculating glitch amplitude and width propagation require these delay values. It is well known that the delay of a gate is a function of the slope and the load capacitance. Hence in our work, we have characterized the delay and slopes of the output for each of the basic cells for different input slopes and for different input combinations by varying load capacitance.

3.2. Flip-Flop Characterization

This characterization is used to determine the *tw* of the flip-flops used in our designs. Our characterization involves sweeping a voltage pulse of a specific width and height at the input of the flip-flop and finding the *tw* during which this pulse is latched by the flip-flop using HSPICE simulation. This characterization is repeated for different pulse widths and heights. The *tw* is expressed as a fraction of the over all clock period. We repeat this experiment for different pulse widths and heights to complete the characterization. The fact that height of the pulse is also used to characterize the *tw* makes this method more accurate. For example, two pulses of same width but different heights might have different *tw*s making our approach more accurate than approximations such used in [1], which assume that only a pulse completely encapsulating the latching window can cause an error.

3.3. Modeling Voltage Glitch Propagation

Next, we propose a set of mathematical equations assuming a triangular or trapezoidal pulse for determining how the voltage pulse amplitude and width vary as they propagate through logic gates towards the flip-flop input.

First, we focus on estimating the amplitude of the output voltage pulse given the input pulse width (PW_i) and the slopes of the output pulse (t_f and t_r). Assuming a linear output slope, for a 1->0 output pulse of an inverter, the minimum output voltage Vo_{min} can be calculated as follows

$$Vo_{min} = \begin{cases} 1 - PW_i / (t_f * 1.25), & PW_i < t_f * 1.25 \\ 0, & PW_i > t_f * 1.25 \end{cases} \quad (1)$$

Here, t_f is the output fall time which was found from the delay and slope table discussed in section 3.1. t_f is the time required for the output changing from 90% to 10% of V_{dd}. Consequently, we use a scaling factor of 1.25 to mimic a complete swing from V_{dd} to zero. Similarly, for a 0->1 output pulse of an inverter, the maximum output voltage Vo_{max} can be given as follows:

$$Vo_{max} = \begin{cases} PW_i / (t_r * 1.25), & PW_i < t_r * 1.25 \\ V_{dd}, & PW_i > t_r * 1.25 \end{cases} \quad (2)$$

Where, t_r is the rise time of the output pulse (from10% to 90% of V_{dd}). The accuracy of the model is directly dependent on the accuracies of the pulse width and the slope values from the pre-characterized tables.

Next, we focus on estimating the width of the output voltage pulse. The pulse width of the output pulse can be modeled as a function of delay of the gate. Approximating the output pulse to be a triangular pulse, we model the output pulse width (PW_o) using the following equation:

$$PW_o = (PW_i - d_1) + X * d_2 \quad (3)$$

Where the delays d_1 and d_2 are the first and second transition delays of the output waveform and

$X = (Vdd/2 - Vo_{min})/(Vdd/2)$ for a 0->1 input pulse and

$X = (Vo_{max} - Vdd/2)/(Vdd/2)$ for a 1->0 input pulse.

This scaling using X is performed because delays (d1 and d2) in the characterization table are determined assuming full voltage swing at 50% switching point (Vdd/2), while actual voltages are swinging only to Vo_{min} or Vo_{max}. Figure 1 can be used to explain the value of X for positive input pulse. As can be seen the distance between the negative edge of input pulse and the positive edge of output pulse is not the delay d_2 but just a fraction of it given by the expression for X.

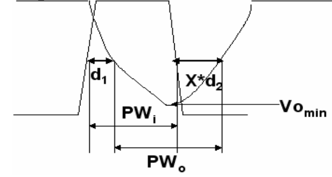


Figure 1. Modeling Pulse Propagation

3.4 Soft Error Estimation Methodology

Figure 2 shows our methodology as applied to a logic chain for a specified set of primary inputs. Here, current pulses are injected in each node. The corresponding voltage pulse is obtained by using the values from a current-voltage (I-V) transfer table. Once a corresponding output voltage is obtained, the propagated pulse width and amplitude at the output of each gate along the path are calculated using the equations presented in the previous sections and the pre-characterized delay models. Since we also account for the state of each node when propagating the pulse, logical masking is accounted for inherently. Once the voltage pulse propagates to the flip-flop, the pulse-width and amplitude values are used to obtain the corresponding *tw* using the flip-flop characterization table explained in section 3.2.

Once the *tw* for a node to one output is known, assuming the probability of a pulse hitting a node N to be P_N , which is a factor of area occupied by the node, the pulse size etc., the soft error rate for the output (for example O in Figure 2), SER_O, can be calculated as follows:

$$SER_O = \sum_N P_N * tw \quad (4)$$

Thus if the circuit has *m* outputs, the overall SER is:

$$SER = \sum_m SER_O \quad (5)$$

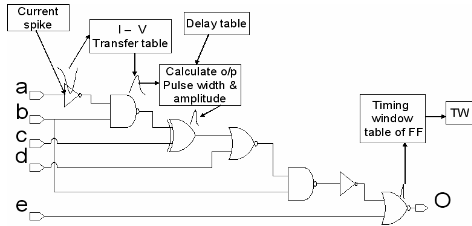


Figure 2. Estimation Methodology

3.5 Re-convergent Paths

The soft error estimation methodology discussed above requires enhancements for handling re-convergent paths. Here, the pulse propagation has to be considered as a special case. There are a couple of important factors that affect the pulse propagation through such paths. The first factor is the input conditions at the re-convergent node which can either result in a magnified or a mitigated pulse at the output re-convergent node. Another important factor is the delay difference between the two paths that lead to the re-convergent gate. This determines the delay of the gate and also determines whether the output pulse can be considered a singe pulse or two different pulses. In our methodology, both these factors can be taken into account by characterizing the cell delays and slopes by varying the time difference between glitches occurring at multiple inputs of the re-convergent gate. After a certain time difference between the arrivals of the glitches, these edges can be considered as separate pulses and hence, the input pulses at the re-convergent gate propagate to the output as two separate pulses. It is to be noted that in both the cases the equations presented in previous sections can be used to obtain the pulse characteristics.

4. SEAT-LA Tool Flow – Implementation

Figure 3 shows the SEAT-LA tool implemented as a part of the bigger tool flow. The tool was implemented using perl and Tcl scripts to work in conjunction with other required tools. As can be seen from Figure 3, the back annotated gate level net-list is taken as an input. Design compiler is used to extract the paths from each node to the output. The tool also requires the capacitance at each node using which the delay and slope tables are to be indexed. These capacitances were obtained from the back annotated net-lists. The state of each node was obtained for a given input vector using the model-sim simulator. Once the state of every node is obtained, SEAT-LA (in Figure. 3) computes the pulse propagation from each node to the output and finds the *tw* as explained in the sections 3.3 and 3.4. This analysis is done for each path of every node. Thus, as described in section 3.5, the SER is obtained by summing up the *tw* for all nodes in the path.

5. Experimental Results

In this section, we present the validation results. First we present the *tw* results of experiments using small designs: ISCAS benchmark - c17, a 4-bit ripple carry adder, a 2x4 decoder and the logic chain shown in Figure 2. All our designs were mapped using the following four pre-characterized cells: an inverter, a 2-input nand, a 2-input nor and a 2-input xor gate and each of the outputs were connected to a Transmission Gate flip-flop. All our pre-characterizations were performed for 70nm Berkley Predictive Technology Model using HSPICE circuit level simulations. We compare the *tw* obtained from our tool with those observed by HSPICE. Next we calculate the error rate based on *tw*s and compare them with HSPICE results. Finally, we present the results of running the tool on bigger ISCAS benchmarks.

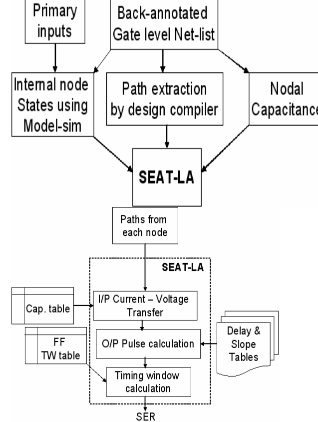
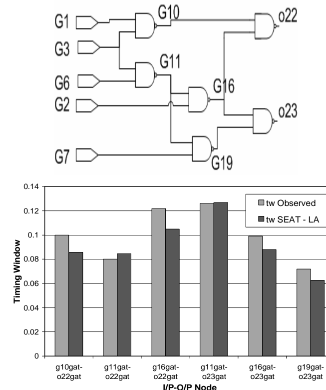


Figure 3. SEAT-LA Tool Flow

5.1 Timing Window Verification

In this section, we present the *tw* of the one of the designs (c17) obtained using SEAT-LA and compare them to observed *tw* from HSPICE simulations. The results for the other small designs are not presented due to space constraints. Each of the smaller designs was implemented using Micromagic, a VLSI layout tool and 70nm BPTM technology. These designs were simulated using HSPICE and the extracted gate level net-list was also given as an input for the tool. Next, the *tw* was experimentally measured using HSPICE simulation by moving the current pulse over a clock period at every node to obtain tw Observed (See Figure 4) for a given input. The timing window (tw SEAT-LA) was calculated by the tool SEAT-LA for the same input for each node using the extracted net-list.



Figue 4. c17 Schematic and Timing Window Results

The *tw* for the ISCAS benchmark c17 are presented in Figure 4. Here the observed and calculated *tw*s differ by a mean error margin of 9.8% and a maximum error of 14.4%. The c17 circuit does have a re-convergent node at o23 as shown in Figure 4. Thus at certain input states, a pulse at G11 can propagate through both nodes G16 and G19 to o23. Since the delays to both the nodes are same, the delay difference between the two pulses is zero. Thus the delay corresponding to both inputs switching is used to calculate the output voltage characteristics by our tool instead of

treating them as two separate glitches at the two different inputs of G16. The *tw* evaluation for node G11 for two different cases illustrates the importance of re-convergent modeling. In Figure 4, the *tw* for a pulse at G11 when it propagates through both G16 and G19 results in a *tw* of 0.127. Now, if the *tw* is calculated assuming that the pulse propagates only through G19, in that case, the value is 0.094.

5.2 Soft Error Rate Verification

In this section, we present our results in verifying the SER from the observed and the calculated *tw* presented in the previous section using equations in section 3.4. The soft error numbers obtained from the equation are compared with values obtained from errors observed at the flip-flop when injecting random errors at the nodes using HSPICE circuit simulation. For this purpose a small test bench was set up to inject random pulses in the nodes of each design. The injection site (node) and time (within one clock period) were chosen randomly by the test bench. A current pulse of given width and height were subsequently injected. The test bench observed the state of the flip-flop and checked for the occurrence of an error. This procedure was iterated 5000 times for each design and the corresponding errors obtained to calculate the "SER Hspice" entries in Table 1. The times required for 5000 iterations are reported in column 6.

Table 1 presents the results for all the above designs along with an inverter chain design. Here, the SER *tw*obs is the SER calculated from the tw observed in Figure 4, SER SEAT-LA is the SER calculated by the tool and SER HSPICE is the SER as obtained from the test bench explained above. The error percentage between SER SEAT-LA and SER Hspice is presented in the last column.

The SEAT-LA results match well with the HSPICE with an average error margin of 6.5% (mean). They also match well with the SER *tw*obs with a mean error of 7.3%. SEAT-LA also has a maximum speed up of 27000 and an average speed up of 15000 over HSPICE simulation.

Design	SER *tw*obs	SER SEAT-LA	Time (min)	SER Hspice	Time (min)	% Error
invchain	0.0733	0.0765	0.01	0.0756	270	1.176
c17	0.1498	0.1382	0.04	0.1410	426	2.027
Decoder	0.0205	0.0234	0.02	0.0210	670	10.120
logicchain	0.0668	0.0649	0.22	0.0620	719	4.399
Adder	0.0355	0.0388	1.4	0.0332	872	14.433

Table 1. Soft Error Rate Comparison

5.3 Experiments on ISCAS Benchmarks

In this section, we present the scalability of our approach using larger ISCAS benchmarks. All the benchmarks used were much bigger than the small designs and hence verification by HSPICE could not be done as in the case of previous designs due to very long simulation times. All the simulations were run on sun-fire-v210 work stations with solaris-unix operating system and 4GB RAM. Table 2 gives the error rate and also the time taken in minutes. We observe that the time required by our tool increases with the number of paths in the design. However, soft errors in combinational logic become more important with shallower pipeline stages. Consequently, the number of paths to be analyzed by our tool for a single combinational logic stage is expected to reduce due to reduced logic depth in a pipeline stage in future. It is also to be noted that since this tool was written using perl and tcl scripts, there can be many optimizations to make the tool work much faster and efficiently than its current status.

6. Conclusion

In this work we have proposed a new methodology to model SER in logic circuits. We have built a logic level tool which takes in a verilog net-list and the parasitic capacitances for the nodes in the net-list as input and gives the SER for the circuit for any given current pulse. We verify this tool and hence the methodology using HSPICE simulations and present our results.

We first verify the *tw* for different nodes in small circuits by showing the *tw*s calculated by our tool match the observed values closely. Next, the soft error rate is verified by using a tool running HSPICE simulations on these designs with random node selection and at random time. The results for the same are presented here and we find that the error margins are around 6.5% (mean) as compared to circuit level simulations with an average speed up of 15000. Next we run the same tool on bigger ISCAS benchmark net-lists and present the SER and time required to run the same.

Circuit Name	Circuit Func.	Total Gates	# of I/Ps	# of O/Ps	SER	Time in min
C432	Priority Decoder	160	36	7	0.0725	108
C499	ECAT	202	41	32	0.0041	216
C880	ALU and Control	383	60	26	0.0188	102
C1355	ECAT	546	41	32	0.0070	162
C1908	ECAT	880	33	25	0.0011	1073
C2670	ALU and control	1193	233	140	0.0034	547

Table 2. SER for ISCAS Benchmarks

7. References

[1] Shivakumar P., M. Kistler, S. Keckler, D. Burger, and L. Alvisi, Modeling the effect of technology trends on the soft error rate of combinational logic., Proc. of International Conf. on Dependable Systems and Networks, June 2002 Pages: 389 – 398.

[2] Zhang M and Shanbhag N. R, A Soft Error Rate Analysis (SERA) Methodology. Proc. Of International Conf. on Computer Aided Design, November, Nov. 2004 Pages: 111 – 118.

[3] Zhao C., Bai X., Dey S., A Scalable Soft Spot Analysis Methodology for Compound Noise Effects in Nano-meter Circuits, 41st Design Automation Conf., June 2004, Pages 894 – 899.

[4] Baze M., Buchner S., Attenuation of Single Event Induced Pulses in CMOS Combinational Logic, IEEE Trans. on Nuclear Science, 44(6), Dec. 1997, Pages:2217 - 2223.

[5] Liu X., Papaefthymiou M.C., A statistical model of input glitch propagation and its application in power macromodeling, The 45th Midwest Symposium on Circuits and Systems, Volume: 1 , 4-7 Aug. 2002 Pages: 380-383.

[6] Chung K. S, Kim T, Lin C.L., G-vector: a new model for glitch analysis, Proceedings of Twelfth Annual IEEE International ASIC/SOC Conference, Sept. 1999 Pages: 159 – 162

[7] Omana M., Papasso G., Rossi D., Metra C., A model for transient fault propagation in combinatorial logic, 9th IEEE On-Line Testing Symposium, 7-9 July 2003 Pages:111 – 115.

[8] Dhillon Y. S., Diril A. U., Chatterjee A, Soft-error tolerance analysis and optimization of nanometer circuits, Proc. of Design, Automation and Test in Europe, Vol. 1, 2005 Pages: 288 - 293.