Project LEARN: Logic Energy and Resiliency

Stevo Bailey
Berkeley Wireless Research Center
UC Berkeley
stevo.bailey@eecs.berkeley.edu

Amudhan Venkatesan
UC Berkeley
amudhan_v@berkeley.edu

I. Introduction

Resiliency is a metric of growing concern as chips scale, and is important for specific applications, like space-based circuits. Specifically, radiation-induced soft errors are not only a problem for large memory caches, but are a growing concern for general flops and combinational logic blocks. Research suggests that below the 45nm technology node, logic soft-error failures are as prevalent as those in unprotected memory arrays [1]. While numerous techniques exist to combat the radiation problem, they all incur substantial overhead to achieve acceptable failure rates. Since energy conservation has arisen as the dominant and primary focus of circuit and processor design, especially as designs become bigger and more complex, it makes sense to explore the energy overhead of different logic soft error resiliency techniques. From these energy vs. failure rate plots, a designer can quickly settle on which resiliency method to adopt to minimize energy for a target FIT[1] rate.

The goal of this project was to produce energy vs. FIT plots for various logic and flop resiliency techniques and various circuit designs. A secondary goal was to scale these calculations to large designs, since accurately estimating energy and FIT can take many CPU hours.

II. Resiliency Techniques

We started by exploring possible logic-level resiliency techniques. The chosen techniques are listed below, where * indicates an omission due to time constraints.

1) **Performance scaling**: sweep the clock period
2) **Flop hardening**: upsize flops or use clever design/layout techniques [3][4]
3) **Gate hardening**: upsize gates
4) ***Redundancy**: triple modular redundancy (TMR)
5) **BISER**: built-in soft error resiliency for flops and combinational logic (CL) [5]
6) ***Datapath ECC**: error-correcting code (e.g. parity [6] or residue [7]) on pipeline and/or arithmetic blocks
7) ***Razor**: timing error resiliency (e.g. [8])

The design was resynthesized for each clock period target. For the other techniques, modifications were made to the post-synthesis Verilog. Flop hardening involved replacing each 1x flop with a 10x flop (see Section IV-B). We implemented flop hardening and the time-shifted outputs versions of BISER.

III. Calculating Energy

Each design's energy was estimated using Synopsys CAD tools. Synopsys VCS and random input vectors were used to find activity factors for every node in the post-synthesis results. Synopsys PrimeTime annotated the design with these activity factors to find the circuit power, from which energy per cycle was calculated. This approach takes hours for large designs (1 million gates) and minutes for small designs.

IV. Calculating FIT

Each design's FIT was estimated using Berkeley FIT (BFIT), a free tool developed at Berkeley to measure combinational logic FIT rates (CFIT) [2]. The tool had to be modified to work with our technology, ST's 28nm FDSOI process.

A. BFIT gate precharacterizations

BFIT relies on the assumption that glitch attenuation for modern technology nodes is negligible. Thus CFIT depends only on logical masking, timing window masking (an erroneous glitch must arrive during the latching window of a flop), and gate properties. Gate properties include sensitive diffusion area, delay, critical charge (the amount of charge needed to produce a glitch big enough to be latched, Q_{crit}), and input capacitance. The critical charge amount depends on arrival time within the clock

[1]FIT = failures in time, or failures per billion hours of operation

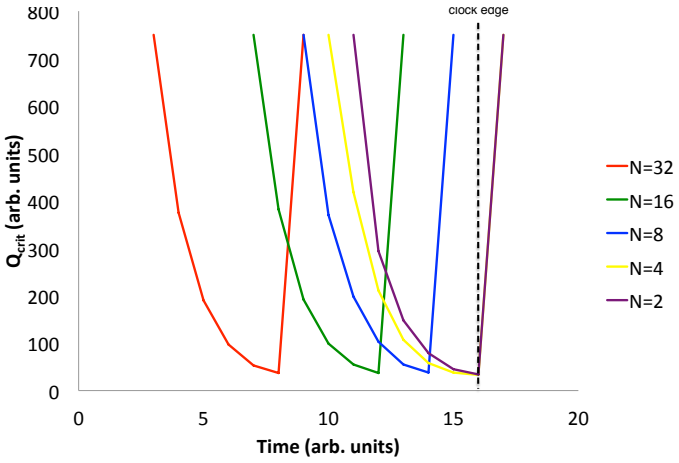

Fig. 1. $Q_{crit}(t)$ for different path lengths; N = number of inverters between the struck gate and the flop; struck gate = minimum NAND2 with input 01

period. The sensitive area was calculated for each possible input vector using a script to parse the gate's SPICE netlist. These gate properties are precharacterized, while BFIT's algorithm handles logical masking and timing window masking.

We tested glitch attenuation for our technology. Figure 1 shows that $Q_{crit}(t)$ is independent of the path length between the struck gate and the flop. While this does not prove electrical attenuation is insignificant for all cases (such as higher fanouts or larger gates), it suggests that we can still use BFIT to obtain conservative yet accurate CFIT estimates.

BFIT also neglects internal gate nodes. Thus non-inverting gates (like AND) have a significant error due to the omission of the internal node. We restrict our design to inverting gates like NAND, NOR, AOI, and OAI for accuracy. This reduces the quality of results from synthesis, but allows much more accurate FIT calculations. Future work will modify BFIT to handle non-inverting and larger gates.

B. CFIT and LFIT

BFIT uses the precharacterizations to calculate CFIT. The algorithm is described in the user's guide, so it will not be detailed here.

CFIT excludes sequential elements, whose contributions (LFIT) can be significantly higher than that of logic elements. To calculate LFIT, we measured Q_{crit} for different nodes and input vectors of two flip-flops (FFs), shown in Figure 2. Since all flip-flops in the technology library had about the same size storage devices (just differently sized output drivers), we created

TABLE I
LFIT CONTRIBUTIONS FOR TWO DFFS

1x DFF	Diff. Area (um^2)	Q_{crit} (fC)	LFIT
Node 1 (0)	0.0546	2.80	1.69e-4
Node 1 (1)	0.0505	1.63	2.10e-4
Node 2 (0)	0.0790	11.28	6.71e-5
Node 2 (1)	0.0727	8.02	2.08e-4
Node 3 (0)	0.0466	2.56	1.49e-4
Node 3 (1)	0.0447	1.65	1.86e-4
Node 4 (0)	0.0091	4.29	2.27e-5
Node 4 (1)	0.0093	2.41	3.69e-5
		Total LFIT	5.24e-4
10x DFF	Diff. Area (um^2)	Q_{crit} (fC)	LFIT
Node 1 (0)	0.2139	27.54	1.46e-5
Node 1 (1)	0.2088	21.03	2.78e-4
Node 2 (0)	0.1502	104.08	8.05e-11
Node 2 (1)	0.1368	67.22	1.20e-5
Node 3 (0)	0.3453	27.11	2.50e-5
Node 3 (1)	0.3162	14.96	6.00e-4
Node 4 (0)	0.0590	41.20	4.96e-7
Node 4 (1)	0.0576	23.89	6.51e-5
		Total LFIT	4.97e-4

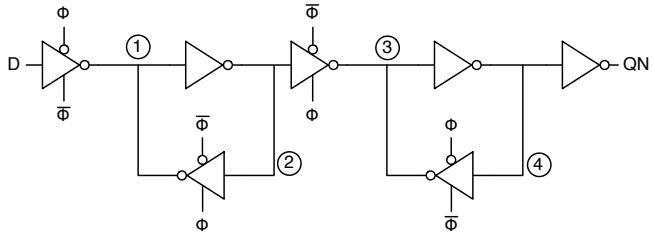

Fig. 2. C^{2}MOS D flip-flop with inverted output

a custom flip-flop with 10x upsized storage devices. For each storage node, we calculated Q_{crit} during the time the node was holding state, then summed up the FIT contributions of each node. Each node is only holding state half the time, so the final LFIT is divided by two. Table I shows sensitive diffusion areas, Q_{crit} values, and LFIT contributes for nominal and 10x flip-flops at each node numbered in Figure 2. For each node, the stored value is shown in parentheses.

Surprisingly, upsizing of this flip-flop only marginally improves LFIT (by 5%). While the Q_{crit} values are about 10x higher, the larger area makes the 10x DFF more vulnerable. Each transistor was upsized by adding 10 fingers, so the area is not exactly 10x bigger. Also, the 1x DFF has a transmission gate between nodes 2 and 3 instead of a clocked inverter, making node 2 much more susceptible, as seen.

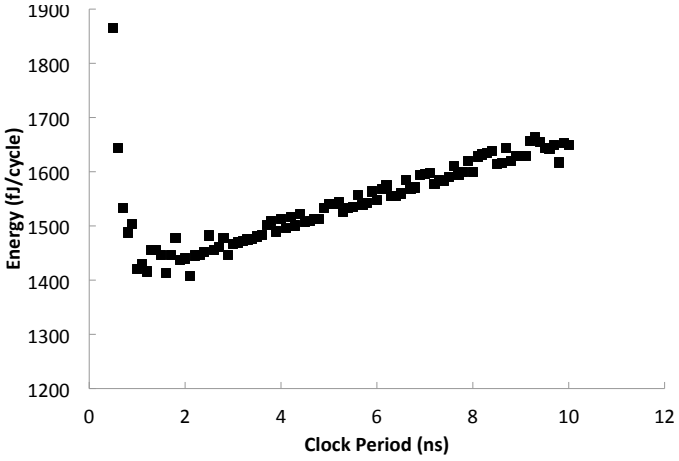

Fig. 3. Energy vs. clock period for the s1238 benchmark

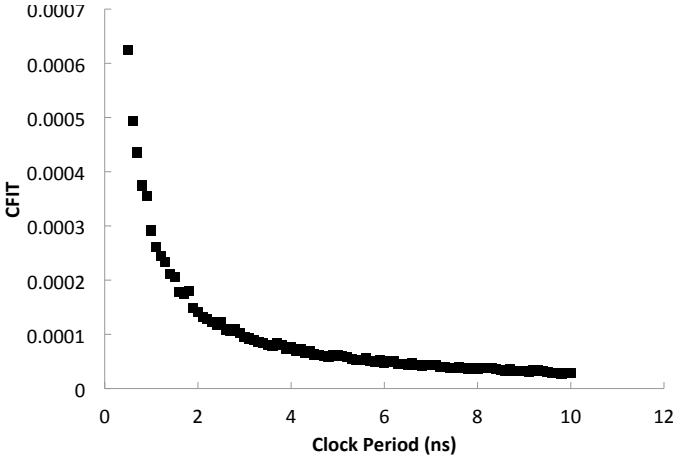

Fig. 4. FIT vs. clock period for the s1238 benchmark

V. Results

We ran simulations with two ISCAS benchmark circuits. The s1238 has about 350 gates and 18 flops, while the s38584 has about 7000 gates and 1200 flops. Both produced similar results, so only the s1238 results are presented here.

A. Performance scaling

As expected, the energy drops dramatically as clock period increases. However, once the design is completely minimum sized, the energy per cycle cannot be lowered further. In fact, with further increases in the clock period, these devices leak more per cycle, causing the energy to linearly increase. Figure 3 shows this trend.

Increasing the clock period gives diminishing FIT reductions. The bigger gates used at low clock periods do not offset the higher susceptibility per cycle since each cycle is short. However, at high clock periods, the FIT decreases because it gets harder to strike the right location at the right time to be latched by the flop. This trend is evinced in Figure 4. We neglected LFIT for this calculation since flop resiliency does not change with the clock period, and so would be a constant offset to the total FIT.

The composite plot of Figure 5 shows a pareto-optimal curve that suggests scaling clock period works until you reach the minimum energy point. After this, further scaling reduces FIT, but leakage causes energy to increase. Circuits hoping to minimize energy and maximize resiliency should start at the minimum energy point. If further resiliency is required, a comparison of techniques would be useful. Thus other resiliency technique simulations were run at a clock period of 2ns for the s1238 benchmark.

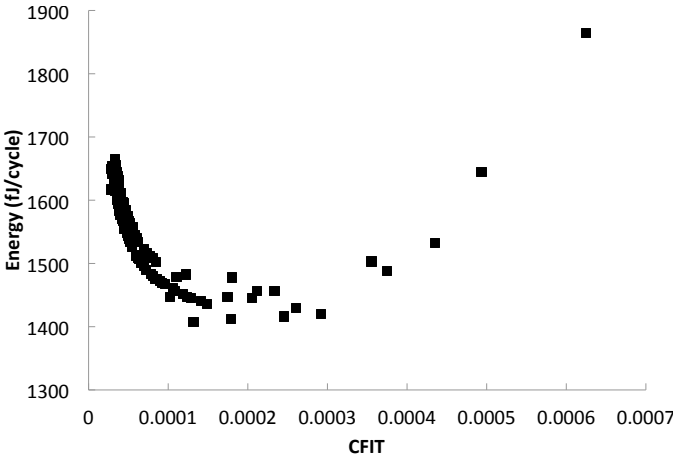

Fig. 5. Energy vs. FIT for the s1238 benchmark as clock period changes

B. Register hardening

As expected from the results in Table I, the FIT decreases as more registers are replaced with the upsized flops. However, the 10x DFF has more capacitance, so it's switching energy is higher. This results in a linear dependence of both FIT and energy on the percent of registers hardened. The combined energy vs. FIT plot is shown in Figure 6 as the percentage of registers hardened in the s1238 benchmark changes. This total FIT combines the combinational logic FIT (CFIT) with the sequential element FIT (LFIT). The random inputs used for both energy and FIT calculations likely contribute to the noise. Note the clock period is 2ns.

C. Gate hardening

Gate hardening involved replacing standard cells with bigger versions of themselves. Since the standard cell library already includes bigger versions in the form of

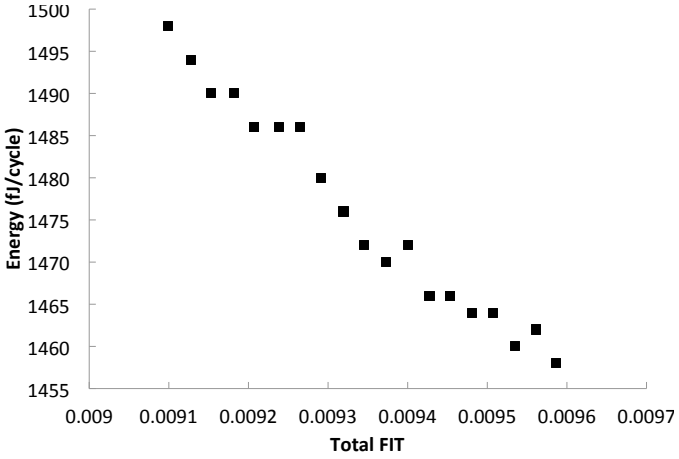

Fig. 6. Energy vs. FIT for the s1238 benchmark as percent registers hardened changes

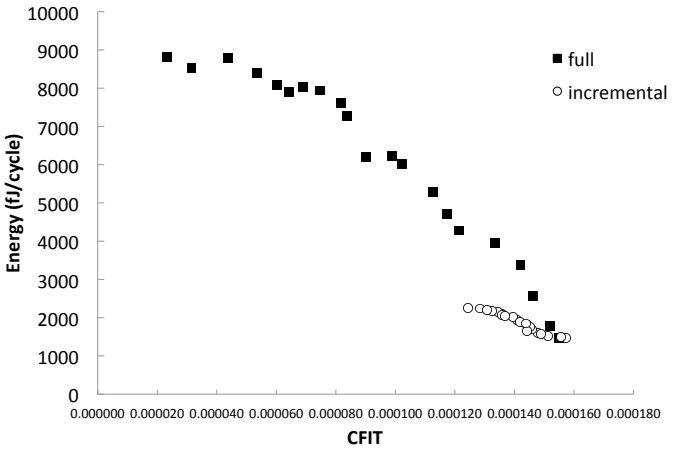

Fig. 7. Energy vs. FIT for the s1238 benchmark as percent gates hardened changes

bigger drive strength cells, those are used. One approach is to replace each cell with the next bigger cell in the library (incremental hardening), while another is replace each cell with the biggest version (full hardening). Both have the same trend as register hardening: FIT and energy are both linear in the amount of hardening. However, bigger gates have a huge impact on both CFIT and energy. Figure 7 shows energy vs. FIT for both full and incremental hardening. Since LFIT is not affected by gate hardening, it is excluded from this plot. The data suggest diminishing returns in gate upsizing. Incrementally hardening all gates before upsizing futher is the best approach.

D. BISER

As presented in [5], BISER has three versions for register-based designs. The first improves LFIT by

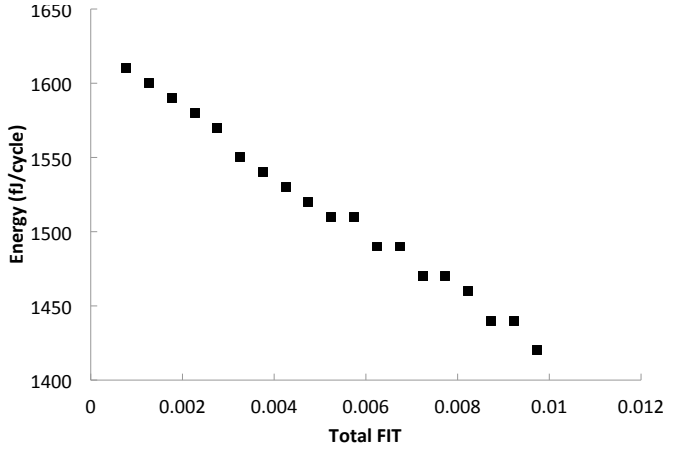

Fig. 8. Energy vs. FIT for the s1238 benchmark as percent registers hardened with BISER-FF changes

adding a C-element to the output of each register, which is doubled. We call this BISER-FF. The second duplicates combinational logic as well, reducing both CFIT and LFIT. We call this BISER-CL. The third delays the output of the combinational logic block by some value τ, and only latches the result when both are equal. We call this BISER-TSO for time-shifted outputs. To simplify analysis, we omitted BISER-CL. We suspect duplicating a combinational logic block will be prohibitively expensive in energy, especially compared to BISER-TSO.

Assuming BISER-FF reduces LFIT for a latch by 20 fold [5], we obtained plot in Figure 8. Note that the energy overhead is largely due to the added register. However, a clever designer can simply reuse scan registers to mitigate this overhead.

BISER-TSO requires adding the extra τ delay to the clock period. However, since we are at the minimum energy point (2ns), small changes in the clock period do not affect energy significantly. We chose a τ of 25ps, or a chain of 6 inverters. Using BFIT to properly measure FIT with BISER-TSO is tricky. A logic soft error can upset multiple registers, so if some of those are unprotected, there will still be a failure. Thus we leave BISER-TSO FIT calculations as future work.

E. All resiliency techniques

To properly compare all techniques, we calculated total FIT for each and combined them onto one plot. Figure 9 shows that clearly latch-based techniques are preferred, since latches contribute a high percentage of the total FIT. Both latch hardening and BISER-FF contribute little extra energy compared to combinational logic resiliency techniques.

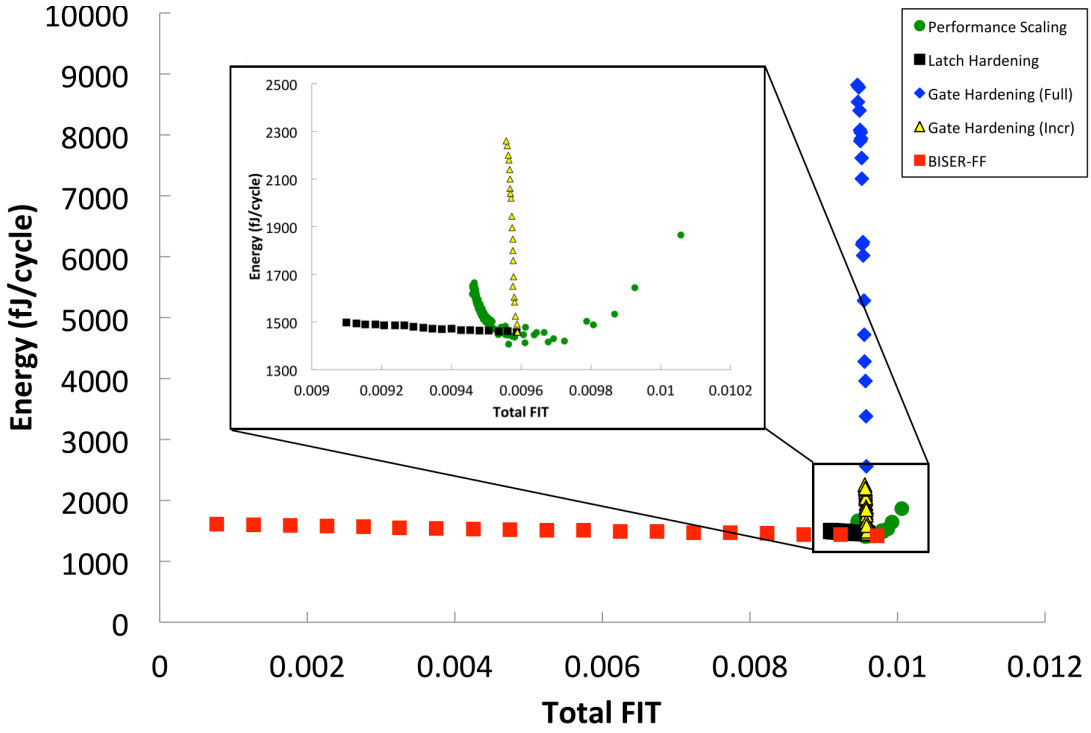

Fig. 9. Energy vs. FIT for the s1238 benchmark for all resiliency techniques explored

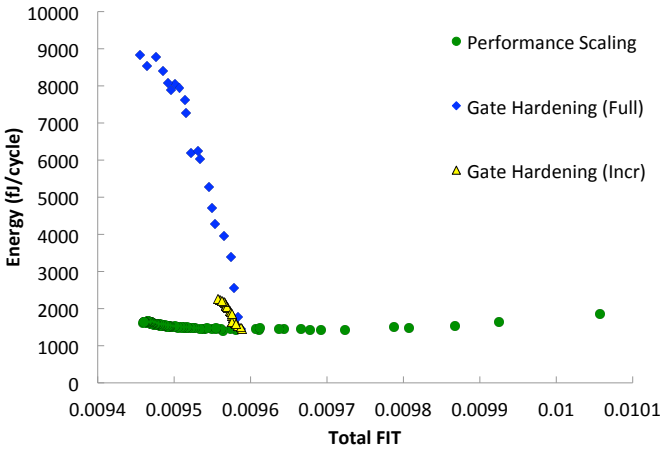

Fig. 10. Energy vs. FIT for the s1238 benchmark for logic resiliency techniques explored

However, a more interesting question is which combinational logic technique is best. Figure 10 shows that upsizing gates has a much larger energy overhead than just slowing down the circuit. If energy is the primary concern, sacrificing performance for FIT is the best choice. If a performance target is also given, then operate as slow as possible, upsizing gates as needed for additional soft-error resiliency.

VI. Conclusion

Several logic soft error resiliency techniques exist. This work explores the energy and resiliency tradeoffs for a subset of them for a given benchmark circuit in ST's 28nm FDSOI technology. Because sequential element FIT is much higher than combinational logic FIT, hardening latches should be the primary concern. If logic resiliency is necessary, operating as slow as possible should precede upsizing gates.

Ideas for future work

- **Scaling algorithms:** scale BFIT and energy calculations for large designs
- **Unexplored techniques:** TMR, Razor, and datapath ECC
- **Vectorless FIT calculation:** Can we calculate or approximate FIT without thousands of input vectors?
- **Automation:** automate LFIT calculation, automate design hardening
- **Improved FIT calculations:** modify BFIT to handle BISER, ECC, etc.

VII. Course feedback

A. Project load distribution

For efficient results, we worked in parallel to setup our respective flows.

Stevo set up Synopsys CAD tools to run simulations and provide accurate energy measurements, while working on modifying BFIT tools to work with both energy flows and our chosen technology. Amudhan was involved in setting up the precharacterization scripts while modifying and developing the BFIT tool for our technology. We both worked on integrating our selected error mitigation methods and models into the FIT calculation algorithm and then ran several simulations for parameteric graphical analysis to compare error propagation values with and without mitigation techniques.

B. Applied class topics

- Sensitization and logical masking: BFIT implements dynamic programming which finds sensitized paths and their delays. The gate precharacterization data characterizes these delays, and combining the above two in the analysis process helps determine which charge causes an error.
- DAG and path based traversals: To initiate analysis, BFIT parses the input circuit netlist into a levelized dag. The randomly selected input vectors are then applied and propagated through the DAG one at a time.

Acknowledgment

Thanks to Professor Seshia for resiliency technique suggestions and introducing us to the BFIT tool. We would also like to thank Dan Holcomb for his support with BFIT as well was Brian Zimmer for his help with the Synopsys tools, specifically SiliconSmart and FineSim for custom liberty file generation.

References

[1] P. Shivakumar, M. Kistler, S. Keckler, D. Burger, and L. Alvisi, *Modeling the effect of technology trends on the soft error rate of combinational logic*, DSN 2002.

[2] D. Holcomb, W. Li, and S. Seshia, *Design as you see FIT: System-level soft error analysis of sequential circuits*, DATE 2009.

[3] S. Lin, Y.-B. Kim, F. Lombardi, *Soft-error hardening designs of nanoscale CMOS latches*, VTS 2009.

[4] H.-H. K. Lee, K. Lilja, M. Bounasser, P. Relangi, I. R. Linscott, U. Inan, and S. Mitra, *Design of sequential logic cell using LEAP: Layout design through error-aware transistor positioning*, SELSE 2010.

[5] S. Mitra, M. Zhang, N. Seifert, TM Mak, and K. S. Kim, *Soft error resilient system design through error correction*, IFIP VLSI-SoC 2006.

[6] S. Mathew, M. Anders, R. Krishnamurthy, and S. Borkar, *A 6.5GHz 54mW 64-bit parity-checking adder for 65nm fault-tolerant microprocessor execution cores*, VLSIC 2007.

[7] D. Lipetz and E. Schwarz*Self checking in current floating-point units*, ARITH 2011.

[8] D. Blaauw, S. Kalaiselvan, K. Lai, W.-H. Ma, S. Pant, C. Tokunaga, S. Das, and D. Bull, *Razor II: In situ error detection and correction for PVT and SER tolerance*, ISSCC 2008.