Modeling Radiation-Induced Soft Errors in Logic and the Overhead of Resiliency Techniques

Steven Bailey
Borivoje Nikolic, Ed.

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2014-233
http://www.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-233.html

December 19, 2014

Copyright © 2014, by the author(s).
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

Acknowledgement

In no particular order, I'd like to thank the following students, faculty, and friends: Garen Der-Khachadourian, Ben Keller, Jackie Leverett, Amanda Pratt, Rachel Hochman, Yongjun Li, Angie Wang, Nathan Narevsky, Pi-Feng Chiu, Jeff Hirschey, Paul Rigge, Brian Zimmer, Yunsup Lee, Jaehwa Kwak, Amudhan Venkatesan, Ruzica Jevtic, Miki Blagojevic, Martin Cochet, Dr. Mircea Stan, Dr. Bora Nikolic, Dr. Krste Asanovic, and Dr. Elad Alon.

Modeling Radiation-Induced Soft Errors in Logic and the Overhead of Resiliency Techniques

by Steven D Bailey

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

Professor Borivoje Nikolić
Research Advisor

Date

* * * * * *

Professor Krste Asanović
Second Reader

Date

Contents

1 Introduction **1**
- 1.1 Problem Statement 2
- 1.2 Scope of Work 2
- 1.3 Related Work 2
- 1.4 Research Contributions 3
- 1.5 Organization 3

2 Radiation Models **4**
- 2.1 Radiation Occurances 5
- 2.2 Charge Generation and Collection 6

3 Logic Error Models **10**
- 3.1 Circuit- and Microarchitecture-Level Error Models 10
  - 3.1.1 Latches 10
  - 3.1.2 Logic Gates 11
- 3.2 Architecture-Level Error Models 14

4 Resiliency Techniques **15**
- 4.1 Circuit-Level Resiliency 15
- 4.2 Microarchitecture-Level Resiliency 16
- 4.3 Architecture-Level Resiliency 18

5 Proposed Model **20**
- 5.1 Circuit-Level Error Models 20
  - 5.1.1 Flip-flops 21
  - 5.1.2 Logic Gates 22
- 5.2 Microarchitecture-Level Error Models 23
  - 5.2.1 Flip-flops 23

5.2.2 Logic Gates 24
5.3 Architecture-Level Error Models 25
5.4 Application-Level Error Models 26
5.5 Resiliency Techniques Tested 26
5.6 FIT Estimation 27
5.7 Energy Estimation 27

6 Model Implementation 29
6.1 Precharacterizations 29
6.2 Number of Test Vectors 34
6.3 Results on ISCAS Benchmarks 34
6.3.1 Performance scaling 35
6.3.2 Register hardening 37
6.3.3 Gate hardening 38
6.3.4 BISER 39
6.3.5 All resiliency techniques 40
6.4 Results on Raven3 41

7 Conclusion 44
7.1 Key Contributions 44
7.2 Future Work 45

List of Figures

2.1 Charge generation and collection at a junction [9]. 6
2.2 Simulated radiation current pulse waveforms at different supply voltages [6]. Note the fast rise time and slow fall time of the current pulse. 7
2.3 FIT rates with various pulses injected. Q is total charge in fC. 9

3.1 Simulation setup for measuring Q_{crit} of a master-slave flip-flop [1]. 11
3.2 Calculting the soft-error rate of a logic chain involves accounting for masking effects [13]. 13

4.1 BISER used in a flip-flop-based design [30]. 16
4.2 BISER used with duplicated combinational logic [30]. 17
4.3 BISER using the time-shifted outputs approach [30]. 17
4.4 64-bit parity-checking sparse tree adder design [31]. 18
4.5 Razor flip-flop for fixing timing errors. 19

5.1 Logic resiliency model for transient errors. 21
5.2 C^2MOS D flip-flop with inverted output. 22
5.3 Complex gates were split into primitives, and each primitive was separately precharacterized. 23
5.4 $Q_{crit}(t)$ for different path lengths; CL = load capacitance on the gate; struck gate = minimum NAND2 with input 01. 24

6.1 NAND2 Q_{crit} depends on the input vector. 30
6.2 Gates are only sensitive within a certain timing window. 30
6.3 NAND2 Q_{crit} depends on the load capacitance. 32
6.4 NAND2 timing window depends on the load capacitance. 32
6.5 NAND2 Q_{crit} depends on the flip-flop used for characterization. 33
6.6 NAND2 timing window depends on the flip-flop used for characterization. . . 33
6.7 Distribution of FIT rates for Raven3 at 1 V and 1 GHz with 500 random test vectors. 34

6.8 Energy vs. clock period for the s1238 benchmark. 35
6.9 FIT vs. clock period for the s1238 benchmark. 36
6.10 Energy vs. FIT for the s1238 benchmark as clock period changes. 36
6.11 Energy vs. FIT for the s1238 benchmark as percent registers hardened changes. Note the graph is fairly flat; the energy varies little. Also note the leftmost point has all registers hardened, and the rightmost point has no registers hardened. 37
6.12 Energy vs. FIT for the s1238 benchmark as percent gates hardened changes. 38
6.13 Energy vs. FIT for the s1238 benchmark as percent registers hardened with BISER-FF changes. 39
6.14 Energy vs. FIT for the s1238 benchmark for all resiliency techniques explored. 40
6.15 Energy vs. FIT for the s1238 benchmark for logic resiliency techniques explored. 41
6.16 Energy vs. FIT for Raven3 as gates are upsized. 42
6.17 Energy vs. FIT for Raven3 as the clock period is adjusted. 43
6.18 Breakdown of combinational logic FIT for Raven3 by functional block. “Other” consists of mostly logic surrounding memories, like the caches and TLB. . . . 43

Chapter 1

Introduction

High-energy particles are known to disrupt correct execution of digital systems when they strike. As scaling drives devices and wires smaller, these disruptions may become worse because the radioactive particles stay the same size and energy. Accurate models are needed to characterize the magnitude, frequency, and significance of radiation-induced errors. Should we worry about soft (transient) errors from radiation? If so, what techniques can we employ to reduce the number of failures due to these errors? What are the overheads incurred by using these techniques?

Different environments have different radiation profiles. On Earth, cosmic rays interact with the atmosphere, producing high-energy neutrons and alpha particles which can disrupt terrestrial chips [1]. Radioactive isotopes in some packaging material emit alpha particles, which may strike the packaged chip. In space, large-scale structures and events create huge variations in particle fluxes. Planetary Van Allen radiation belts are rife with radiation and create strong spacial particle variations during a satellite's mission. Solar flares and coronal mass ejections produce temporary but highly-energized particle events. Designers should design chips to deal with the relevant radiation profile. Thus flexible methods for measuring soft error sensitivity and hardening the chip are needed.

1.1 Problem Statement

Current radiation-induced soft error modeling efforts span wide ranges of complexities, hierarchical depths, and accuracies. Additionally, numerous resiliency techniques exist which vary in hierarchical scope, overhead, and effectiveness. This project seeks to unify resiliency models and the overhead of resiliency techniques in terms of area and energy into one coherent framework.

1.2 Scope of Work

This project starts at the circuit-level of abstraction, modling device properties and radiation-induced transients with simple HSPICE models and exponential curves, respectively. It considers combinational logic gates and sequential registers, but omits SRAM. An existing circuit-level failure rate calculation tool, the Berkeley FIT Estimation Tool (BFIT), is modified to calculate the resiliency of logic circuits and flip-flops with and without resiliency techniques for the 28nm FDSOI technology from STMicroelectronics (ST). Higher hierarchical level models are discussed briefly. Synopsys tools find energy, area, and delay metrics for selected resiliency techniques. Simple ISCAS benchmark circuits are used to evaluate the model and correlate it with device-level HSPICE results. Finally, the model's evaluation is scaled up to a realistic microprocessor, Raven3. Raven3 has one combined RISC-V ISA scalar and vector core and is designed to be a low-power mobile processor with DVFS and on-chip switched-capacitor voltage regulators.

1.3 Related Work

Seifert gives a broad and relatively deep overview of radiation-induced soft errors up to the architecture level [1]. He discusses both logic and memory modeling. Several papers discuss specific modeling efforts from the device and physics levels up through the architecture level. Uznanski from ST evaluates their Tool suite for rAdiation Reliability Assessment (TIARA)

by combining it with Geant4, a particle simulator from CERN, and comparing the results on flip-flops and memories with silicon data [2, 3, 4, 5]. Ramanarayanan from Intel presents a Soft-Error Analysis Toolset (SEAT), which includes both device-level and logic-level tools used to estimate the resiliency of designs with hundreds of gates in hundreds of minutes [6]. [mcoracle] These tools are all based on Monte Carlo simulation methods.

1.4 Research Contributions

This work approaches a familiar problem from a new angle. Often resiliency techniques try to globally or locally reduce failure rates to some value. Other times, the resiliency techniques are applied without quantifying their benefits or overhead, but thereafter marking the chip as “rad-hard”. However, there are now many possible techniques, and this work seeks to explore them from an energy perspective. Given a target FIT rate, how can we meet this target while incurring the smallest energy overhead? Each resiliency technique improves FIT rate by some amount, but also increases energy consumption. We quantify both these values using both custom and commercial software to optimally harden a microprocessor.

1.5 Organization

This thesis is organized like the chip design hierarchy. It starts at the physical and device level, then proceeds to discuss resiliency at the circuit, architecture, and application levels. Existing models are discussed first, followed by the proposed model and the model's evaluation.

Note that throughout the paper,“physical mixed-mode simulation” refers to a combination of 3D TCAD and SPICE simulations, while “electrical mixed-mode simulation” refers to a combination of SPICE and Verilog simulations.

Chapter 2

Radiation Models

High-energy particles striking a chip on Earth typically originates from cosmic rays interacting with atmospheric particles or from radioactive isotopes on the chip's packaging [1]. When these neutrons or alpha particles strike a junction, they generate electron-hole pairs. The electron-hole pairs drift and diffuse at the junction, creating a current. However, the generated electron-hole pairs are quickly used up, so the current simply pulses for a short time [6]. This current charges or discharges the struck node. This produces a single-event upset (SEU) if the struck node is a storage node (flip-flop or SRAM bitcell) and the striking particle imparts enough energy to flip the bit. Alternatively, in general logic, this produces a single event transient (SET) which propagates through the logic until it reaches a latch or flip-flop. If the storage element latches the transient pulse instead of the correct data, the microarchitectural state is corrupted and a single-event upset (SEU) occurs [7]. In this section, I present a brief overview of prior work in device characterization of SETs, as well as the model used in the project.

Modeling SETs at the device level can be considered in two parts. The first part describes the frequency of various types of radiation strikes. The second details both the amount of charge imparted by each strike and the nature of the current pulse resulting from this charge.

2.1 Radiation Occurances

Radiation environments vary widely, and any model will depend strongly on the location. Terrestrial environments benefit from Earth's magnetosphere, which blocks most ionizing radiation originating outside Earth. Leaving low-Earth orbit requires passing through the van Allen radiation belts, which store a large number of particles and pose a problem for spacecraft. In outer space, an abundance of high-energy particles come from both our Sun and outside our solar system. Other planets have similar radiation belts to our own, but many have little to no inherent shielding. To simplify modeling efforts, this project assumes a typical terrestrial environment. Flux values come from data collected in New York City. However, this is just a multiplicative factor, so it is trivial to adjust at any point in the model.

From [8], the probability of a particle striking an area per second of at least charge q is given by:

$$R(q) = F \times A \times K \times exp(-\frac{q}{Q_s}) \tag{2.1}$$

where

- F = neutron flux
- A = sensitive diffusion area of a given node for a given node value
- K = technology-independent fitting parameter (see [8]), 2.2×10^{-5}
- Q_s = technology-dependent charge collection parameter, estimated from [8]

Multiplying this probability by the time period of interest gives the probability of observing a particle strike exceeding charge q striking the sensitive node. This equation allows us to translate critical charge values into FIT. Note that this model only includes atmospheric neutrons.

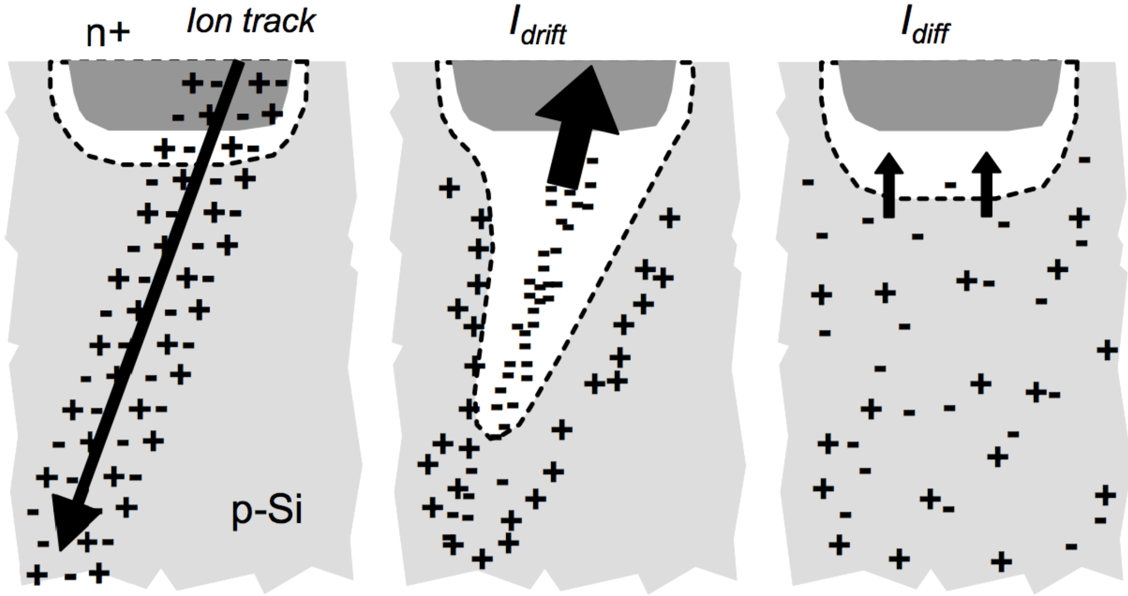

Figure 2.1: Charge generation and collection at a junction [9].

2.2 Charge Generation and Collection

When a particle strikes a junction, it produces a funnel of electron-hole pairs, as seen in Figure 2.1. These electron-hole pairs drift and diffuse, causing a current pulse. Initially, the electric field inherent in reverse-biased junctions separates the charges, spiking the current. Any remaining carriers then diffuse, creating a long tail current in the current pulse. In bulk technologies, parasitic bipolar devices can turn on and amplify the current glitch. Fully depleted silicon on insulator (FDSOI) eliminates this problem.

Many pulse models use a double exponential equation to model the fast spike and slow fall of the current pulse [10]:

$$I(t) = \frac{Q_{coll}}{\tau_a - \tau_b}\left(e^{\frac{-t}{\tau_a}} - e^{\frac{-t}{\tau_b}}\right), \tag{2.2}$$

where Q_{coll} is the total collected charge, and τ_a and τ_b are drift and diffusion time constants. All three of these parameters depend on the technology, voltage, circuit, temperature, and particle energy, so each must be characterized for the particular application and operating condition. This project focuses on a single technology, and characterizes the gates for a

single voltage and temperature. Figure 2.2 shows sample current pulses in a 130nm bulk CMOS process. Another pulse model uses a square root current pulse, which has a similar

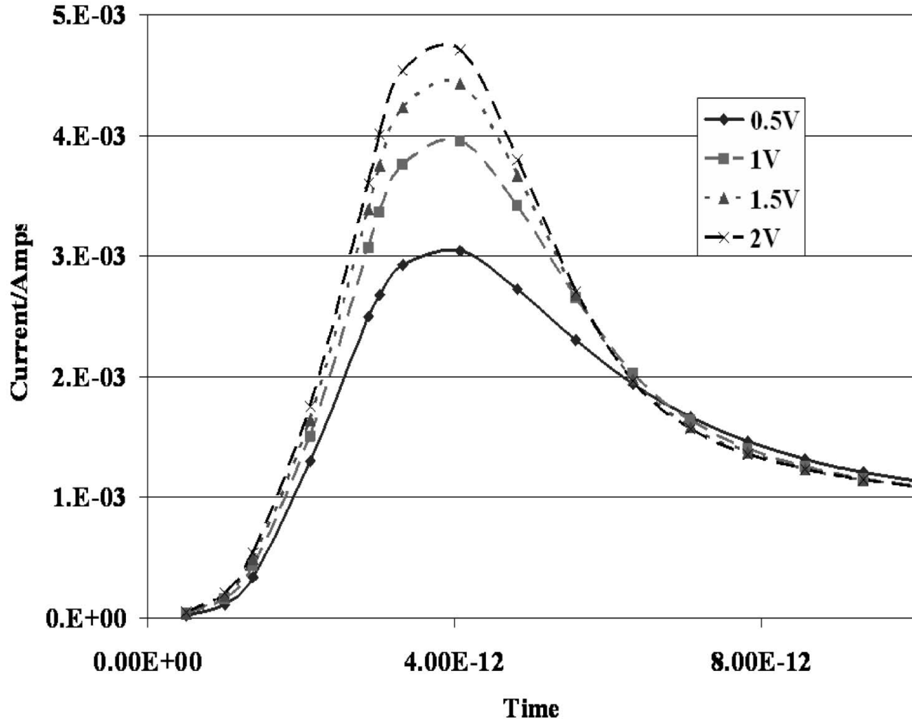

Figure 2.2: Simulated radiation current pulse waveforms at different supply voltages [6]. Note the fast rise time and slow fall time of the current pulse.

shape but slightly different equation [11]:

$$I(t) = KQ_{coll}\sqrt{t}\frac{e^{-t}}{L} \tag{2.3}$$

Here, K is a unitless normalization constant, t is time per unit L, and L is the pulse time constant. While both equations produce similar pulse shapes, the double exponential captures the actual physical response by separating the initial drift and later diffusion currents.

The shape and magnitude of the pulse depends on the technology, incident particle, particle's energy, reaction byproducts, strike's distance from the device, supply voltage, device's polarity, and incident angle. Silicon on insulator (SOI) technologies collect less charge than bulk technologies [12]. For fully depleted SOI devices, there are no parasitic junction diodes since the junctions touch a buried oxide insulator on the bottom and a depleted channel on the side. For both bulk and SOI technologies, neutrons and alpha

particles of various energies strike the silicon and produce an array of reaction products. Particle energies are handled by Equation 2.1, since the charge imparted, q, is a function of particle energy. Different byproducts produce current pulse magnitudes which can vary by up to 200% for a given particle energy [6]. The collected current falls off sharply as the particle strikes further from the transistor junction of interest, so this is often ignored [6]. Increasing supply voltage increases the amount of charge collected, since it boosts the electric field [6]. NMOS devices collect 3x the current of PMOS devices, but this is due to doping differences [6], so SOI devices likely have similar charge collection parameters, Q_s in Equation 2.1. Much prior work has modeled current pulses in various technologies [13] [14] [15]. Because we lack process-specific charge collection information, we use these models to determine the size and duration of representative current pulses for our circuit-level model. Precharacterization involves sweeping Q_{coll} so that different particle energies may be considered. Note that these models rely on TCAD device-level simulation, since measuring the actual parasitic current pulse poses a tough challenge.

While the shape of the current pulse used in simulation was adopted from literature, the actual pulse parameters vary with the operating condition, distance from diffusion, strike angle, and particle energy. These dependencies require device-level simulations to accurately characterize the pulse shape. To model this, the collected charge and time constants from Equation 2.2 were varied. Figure 2.3 shows how pulse parameters change the FIT rate. A SPICE-level simulation setup tested various pulse shapes on a synthesized decoder circuit. When the collected charge Q_{coll} is large, increasing the pulse duration increases the FIT because a significant voltage transient lasts longer. As the collected charge decreases, electrical masking lowers the FIT rate of longer pulses. The pulse dimensions clearly have a large impact on the overall FIT rate. While this work considers total collected charge, it uses a single pulse shape (rise and fall time) for all simulations. More accurate pulse measurements or simulations from 3D TCAD tools would improve the accuracy of the model described in this work.

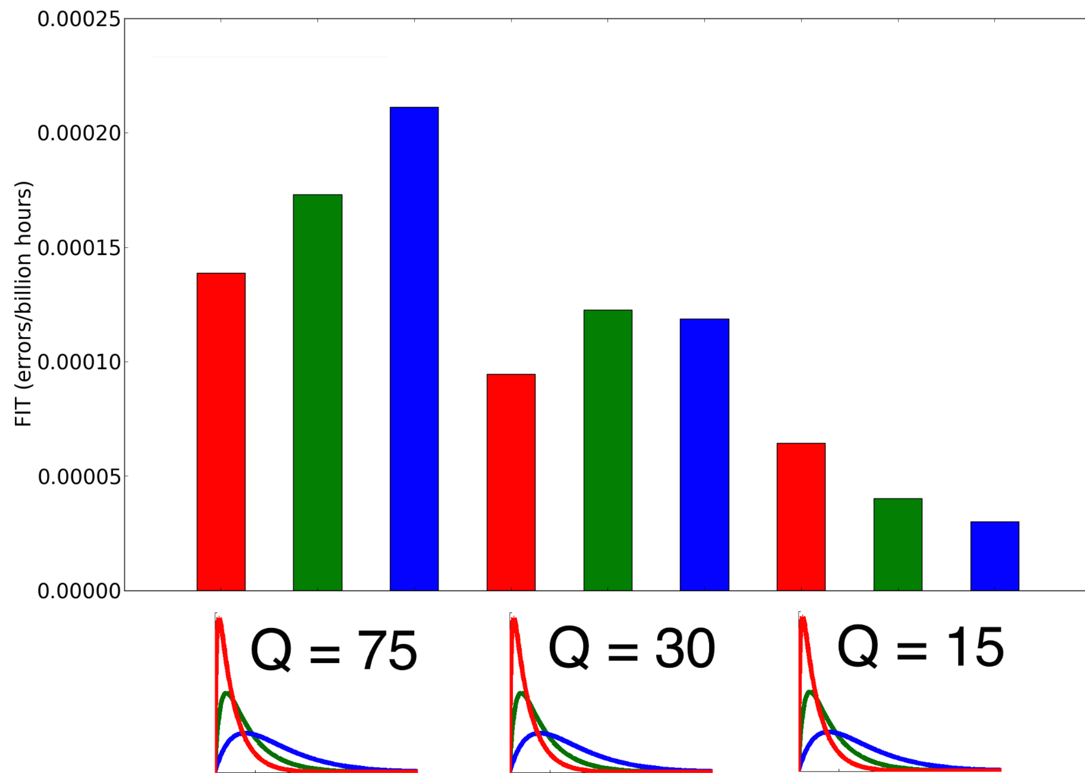

Figure 2.3: FIT rates with various pulses injected. Q is total charge in fC.

Chapter 3

Logic Error Models

3.1 Circuit- and Microarchitecture-Level Error Models

Circuit-level models generally rely on a pulse model to describe the physical charge collection process, as described in the previous chapter. This allows the use of SPICE instead of expensive 3D TCAD models or physical mixed-mode simulation. The general strategy is to inject a current pulse at a certain time and on some node of interest in the SPICE netlist, then run a transient simulation and see what happens. If a latch stores the wrong value on some cycle, an error has occurred. Logic error models tend to focus on registers, since registers contribute the majority of soft errors in general logic [16, 17]. However, combinational gate models also exist and are discussed here.

3.1.1 Latches

For latches, the simulation approach is similar to that of SRAM bitcells, as discussed by Freeman [11]. A pulse is injected at the storage node, and the output is monitored to see if it changes. The pulse properties, in particular total charge, can be varied to find Q_{crit} of the node for some operating condition. The operating conditions, such as voltage, stored value, and input vectors, can also be swept. Figure 3.1 shows a typical setup for testing a

master-slave flip-flop. Because this model is simple, most have adopted it for latch soft error characterization. However, since this model requires just a few transistors, some researchers opt for 3D structural simulations to improve accuracy at the cost of simulation time [18]. Note that it's possible for the high-energy particle to scatter when traveling through the metal stack before striking the latch, if the chip is oriented correctly. This may produce multiple glitches, but the collected charge per glitch will be less because of energy conservation. To simplify modeling efforts, this effect is ignored.

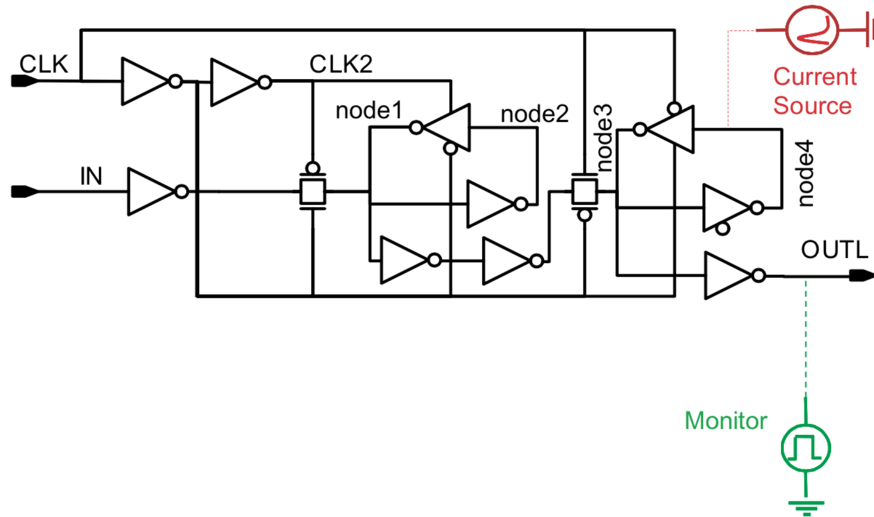

Figure 3.1: Simulation setup for measuring Q_{crit} of a master-slave flip-flop [1].

3.1.2 Logic Gates

Combinational logic gates themselves do not store state, so if one gets struck with a high-energy particle, it only counts as an error if downstream flip-flops latch the wrong value. The strike produces a glitch, which must propagate through the datapath and arrive at the flip-flop with enough amplitude to be latched. The gates only contribute to the failure rates of downstream flip-flops, so ignoring them just harms the accuracy of latch FIT rates. Also, a single gate failure can cause multiple flip-flops to fail. At the microarchitecture level, multiple flip-flops failing are just multiple failures. However, at the global architecture and application levels, multiple failures could be more or less significant than single failures, depending on their location and timing.

There are three factors which combine to mask soft error glitches in logic.

1. **Logical masking** - If the glitch reaches the input of a gate, but that input does not affect the gate's output, the glitch is logically masked.
2. **Electrical masking** - The glitch may attenuate, eventually disappearing, as it propagates through the datapath.
3. **Latching window masking** - If the glitch arrives at the flip-flop outside its latching window, the glitch will not be latched.

These factors make it difficult to determe the failure rate of a logic gate within some circuit. Logical masking requires input vectors or data, but can be simulated in Verilog. Electrical masking requires SPICE simulation. Latching window masking requires timing-accurate Verilog or SPICE simulation. The SPICE simulations pose the biggest hurdle, as they run much slower than Verilog simulations. Different models have proposed different simplifications for handling each masking effect.

Logical masking

Logical masking occurs when a gate's input does not sensitize its output. For example, when a NAND gate has the inputs 10, a glitch that switches the '0' input to a '1' does not change the output of the gate, so the glitch disappears. Most models perform logical masking inherently, since they run gate-level simulation in some fashion to account for electrical masking. Input vectors can either come from simulation, random variables, or probabilities. An important consideration with any logic simulation is reconvergent paths. Reconvergent paths are those which diverge at the output of some gate, then later converge on some other gate downstream. This causes two timing paths to begin at some gate and end on the same flip-flop. Often the worst-case path is used while the other is ignored. This introduces an error in the FIT calculation, the magnitude of which is dependent on the number of reconvergent paths in the design.

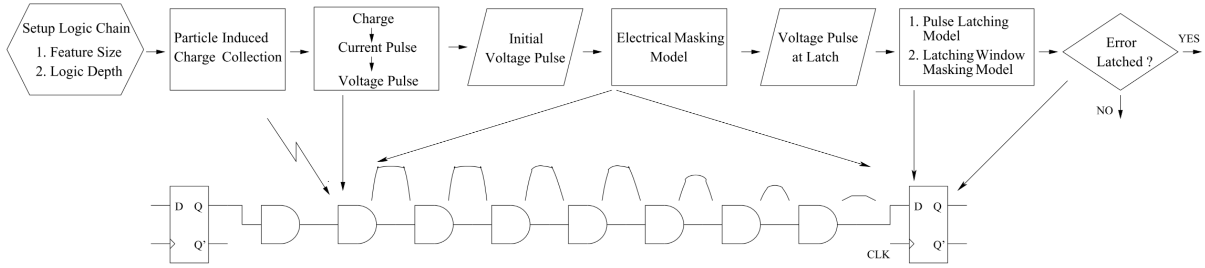

Figure 3.2: Calculting the soft-error rate of a logic chain involves accounting for masking effects [13].

Electrical masking

Short pulses and glitches change as they progress through a circuit. Different delay conditions, including asymmetric ones which affect the rising edge and falling edge differently, modify the rise time, fall time, and width of these glitches as shown in Figure 3.2. One approach to modeling this is to use dynamic electrical mixed-mode simulation, where the circuits surrounding the glitch are electrically simulated in SPICE while the rest of the circuit benefits from simple Verilog simulation [19]. Another approach is to use custom simulation which abstracts many electrical properties of gates. For example, Shivakumar [13] combine a Liberty NLDM-like Horowitz model [20] with a delay degradation model to estimate how the pulse shape changes through a chain of gates. Ramanarayanan [6] has a similar approach, but he simply precharacterizes gates by simulating representative current pulses at the input of a gate and measures how it propagates through the gate under different load conditions. Then, he approximates the pulse as a trapezoid and mathematically approximates how it passes through gates, which obviates the need for SPICE but is more accurate than perfect propagation. A third, conservative approach would be to ignore electrical masking and assume glitches propagate perfectly, and that they arrive at the sequential element with full amplitude.

Latching window masking

The glitch must be latched to cause an error. Some models assume whenever the pulse arrives completely during the latching window, a error occurs [13]. Others simulate different pulses at the input and check whether the correct value was latched or not [6]. Seifert defines a Timing Vulnerability Factor (TVF) which derates the nominal SER to account for timing windows [21]. Instead of directly calculating the timing windows, he calculates the nominal SER without considering timing, then simulates in SPICE to find the actual SER of the circuit. The ratio of these gives the TVF for the circuit. This is a higher-level approach, since it accounts for all timing windows at once instead of each window per flip-flop.

3.2 Architecture-Level Error Models

Circuit-level models produce FIT rates for each flip-flop in a design. Architecture-level models can take advantage of this abstraction and start by assuming each flip-flop has some probability of failure. For example, architecture vulnerability factors (AVF) give the percentage of time a particular bit matters [22]. A bit "matters" if it affects some boundary or the final output of a program. This boundary could be an architecture boundary, like the word written to memory at the write back pipeline stage, or a benchmark boundary, like the output values stored in memory after the benchmark finishes. This modeling method obviates the need for injecting faults. Since it knows when a flip-flop is sensitive, it multiplies that by the probability a soft error occurs to obtain an architecture-level FIT rate.

Instead of finding when a flip-flop is sensitive, another model, called statistical fault injection, injects errors at various points in the transient simulation [23]. It then propagates the error to find if this fault "matters" or not. While stastical fault injection is arguably more accurate than using AVFs, it requires a larger number of Monte Carlo simulations [22].

Either model suggests microarchitecture-level FIT should be derated by about 10% when running common benchmarks on typical processors (see [22] and [23]). Thus not propagating the FIT up through the architecture can give grossly conservative failure rates.

Chapter 4

Resiliency Techniques

The models discussed in the previous chapter are useful for calculting the failure rate of particular gates, circuits, and architectures, but what if they suggest a failure rate higher than acceptable? Many researchers are exploring a variety of techniques to reduce the failure rate at different levels of the hierarchy. When considering resiliency techniques, the implementation overhead is most important. Each technique trades off energy, area, and delay for a lower failure rate, but minimizing the overhead and maximizing the resiliency are nontrivial tasks. This section will explore a variety of resiliency techniques at different levels of the chip hierarchy, including discussions on overhead and optimization.

4.1 Circuit-Level Resiliency

Individual gates and flip-flops can be hardened through redundancy, clever layout, resizing, and circuit tricks. Logic gates can be resized in both dimensions or have their V_{TH} values adjusted to improve reliability [24]. There are no other obvious options for hardening gates at the gate level, but the latch and flip-flop hardening design space has been thoroughly researched.

Since flip-flops comprise more than 90% of sequential logic FIT [16] and store state,

hardening them is both easier and more worthwhile. At a low level, playing with layout topologies may improve the resiliency of flip-flops [25]. However, this comes at a high, 40%, area cost. A common ciruit technique is to use dual interlocked cells (DICE) to interleave feedback between the pull-up and pull-down networks. Wang and Gong [26] improve upon DICE cells by delaying the actual latching edge when struck, so with proper hold and setup time margining the design is resilient to errors. Another common approach is to increase the capacitance on critical nodes. While this has obvious energy and delay penalties, it increases the Q_{crit} required to flip a storage node. Schmitt trigger latches (SEM-latch) rely on hysteresis to quell particle transients, but this does not stop large transients from causing errors [27, 28]. Finally, several redundant latches and flip-flops have been proposed. A local triple modular redundancy (TMR) scheme followed by a majority gate is an obvious choice. Another is called built-in soft error resiliency (BISER) and doubles the latch in combination with a C-element and keeper to choose the output [29].

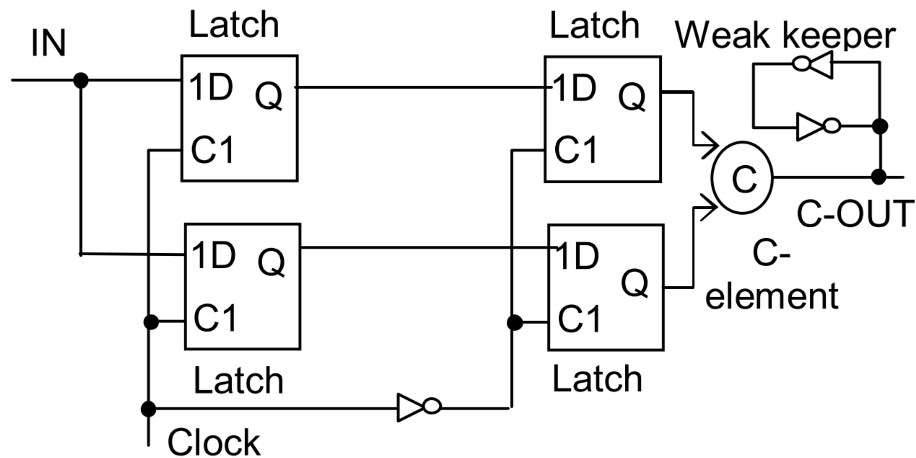

Figure 4.1: BISER used in a flip-flop-based design [30].

4.2 Microarchitecture-Level Resiliency

Small circuits have several methods for resiliency. Redundancy is the obvious option. Triple modular redundancy (TMR) replicates the computation three times, then uses a majority gate to choose the correct answer. If one value is incorrect while the other two are correct, only the correct answer will pass through. This provides single error correction. Mitra uses

the C-element discussed in the previous section to reduce the reduandancy overhead [30]. Using just one extra copy of the combinational logic, BISER can correct single bit errors by waiting until the correct value arrives at the latch, as seen in Figure 4.2. This may require adjusting the clock period, but if the path is not critical, there is no performance overhead. An alternative to duplicating the logic is to delay the result, and directly compare the two values. Mitra calls this the "time-shifted outputs" resiliency technique, seen in Figure 4.3. This method requires increasing the clock period if the delay element lies on a critical path.

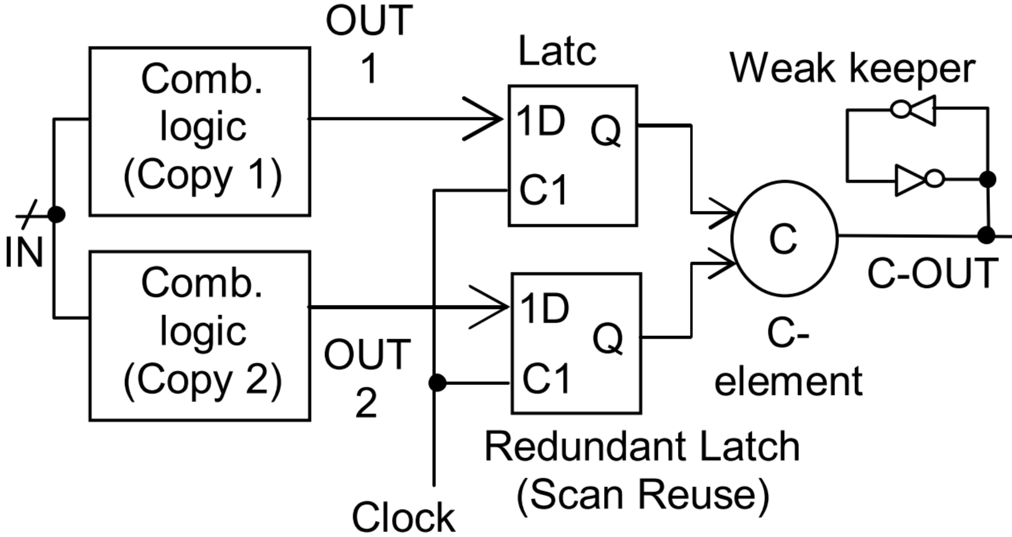

Figure 4.2: BISER used with duplicated combinational logic [30].

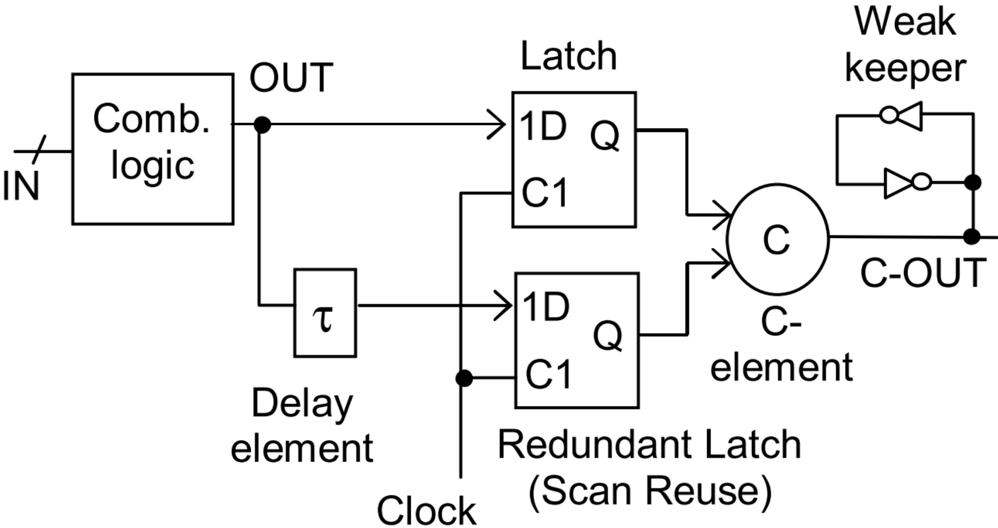

Figure 4.3: BISER using the time-shifted outputs approach [30].

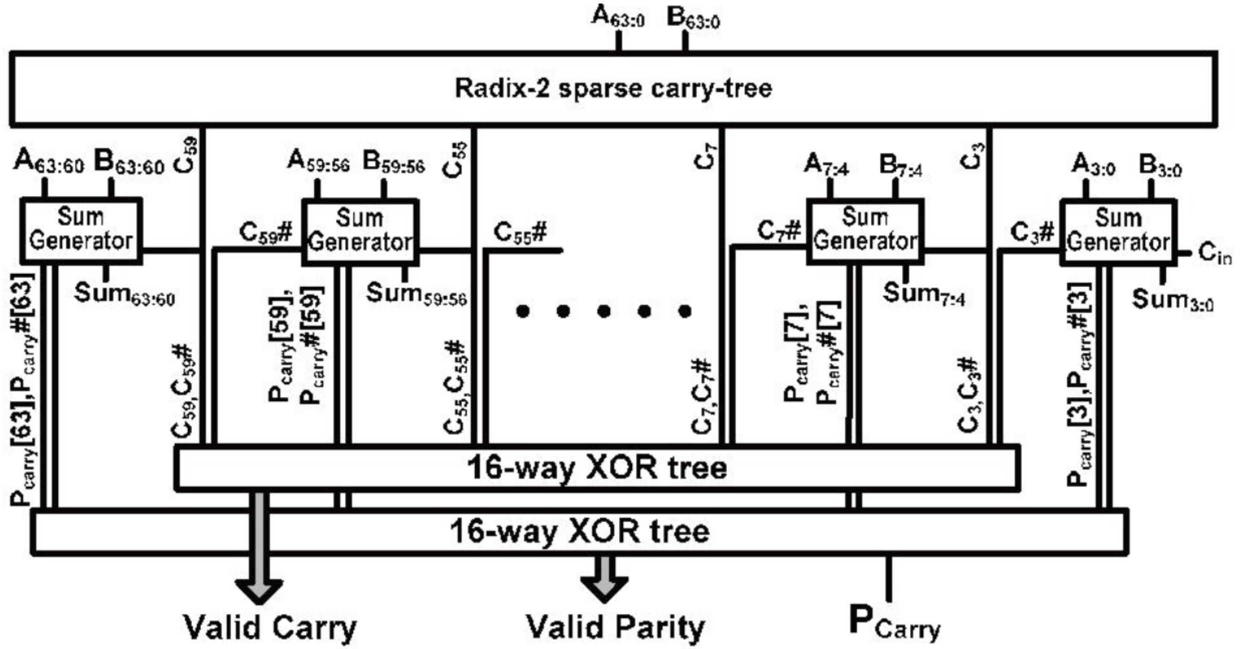

Figure 4.4: 64-bit parity-checking sparse tree adder design [31].

Parity and error-correction codes (ECC) are available at the microarchitecture level. These clever tricks avoid the huge overhead of redundancy by encoding data and selectively duplicating hardware. For example, Mathew presents a parity-checking adder to handle SETs, even those which fanout to and disturb a large number of output bits [31]. Redundant carries and a parity-checking XOR tree are used to detect and correct errors when they occur, as shown in Figure 4.4. ECC is commonly used in SRAMs to detect and correct bit errors, but can also be used for buses of registers.

Finally, adjusting operating conditions can also improve failure rates. Simply running the microarchitecture at a higher voltage or slower clock period can also reduce the failure rate, as discussed later.

4.3 Architecture-Level Resiliency

At the architecture-level, we consider general purpose processors, but the resiliency techniques may extend to some graphics processors, multicore processors, or digital signal proces-

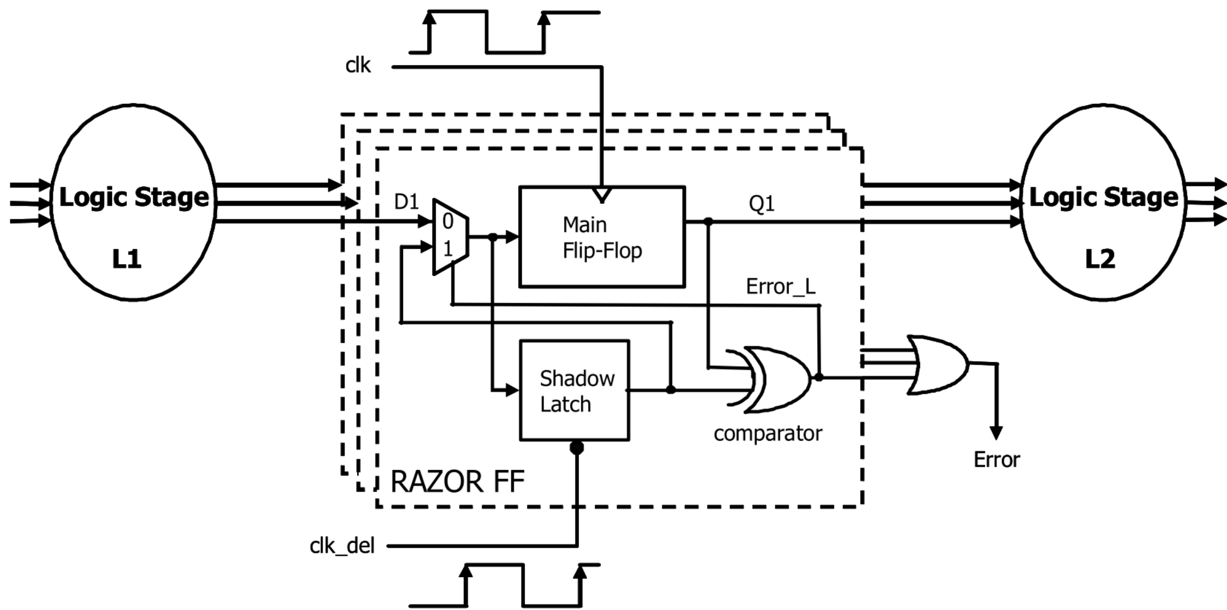

Figure 4.5: Razor flip-flop for fixing timing errors.

sors. Application-specific integrated circuits (ASICs) likely require custom architecture-level resiliency. Just like at the microarchitecture level, ECC can be used in places like the register file or pipeline registers. However, at the architecture level, resiliency can take advantage of the architecture's capabilities to reduce overhead. For example, parity checking can be done in parallel with normal computation. Then when an error is detected, the parity checker stalls the pipeline to recompute or tells the processor to flush the pipeline and start over. This reduces the instructions per cycle (IPC) count, but only when an error occurs, which in typical processors may be negligible.

A similar approach is Razor, which does not encode the data but rather checks for glitches being latched near the clock edge [32]. With Razor, a "shadow latch" stores the input of the main flip-flop at a delayed time, with the delay set by a separate clock, as seen in Figure 4.5. The outputs of the main flip-flop and the shadow latch are compared, with a difference indicating an error. This increases the hold-time margin of the combined storage device, called a Razor flip-flop. With both Razor and ECC mentioned in the last paragraph, the resultant error signal poses a significant overhead. Every error signal from every protected flip-flop must propagate through a large OR tree. Different iterations of Razor have attempted to reduce the overhead of the shadow latch and OR tree [33, 34].

Chapter 5

Proposed Model

The presented resiliency model is split into two main categories: sequential (flip-flops) and combinational (logic gates). These models are cominbed at the microarchitecture level to find an error rate, and the error rate is propagated up the hierarchy through the architecture and application levels. At each level, important design metrics, such as energy/op, area, and performance, are calculated. Figure 5.1 gives a block-diagram view of the proposed model hierarchy.

5.1 Circuit-Level Error Models

Circuit-level error models are independent of circuit microarchitecture and particle strike. Each flip-flop and gate is precharacterized to find the critical amount of charge required to flip the stored bit (Q_{crit}), the diffusion area sensitive to a strike, and other standard circuit-level parameters like delay and capacitance. These values are collected for all possible input vectors and states since they are data-dependent parameters. Simulation is performed using SPICE and transistor-level cell models. Circuit-level resiliency techniques, such as device upsizing, are included here by characterizing gates and flip-flops of different sizes. Since actual gate FIT contributions depend strongly on their location in the microarchitecture, FIT cannot be calculated at the circuit/device level.

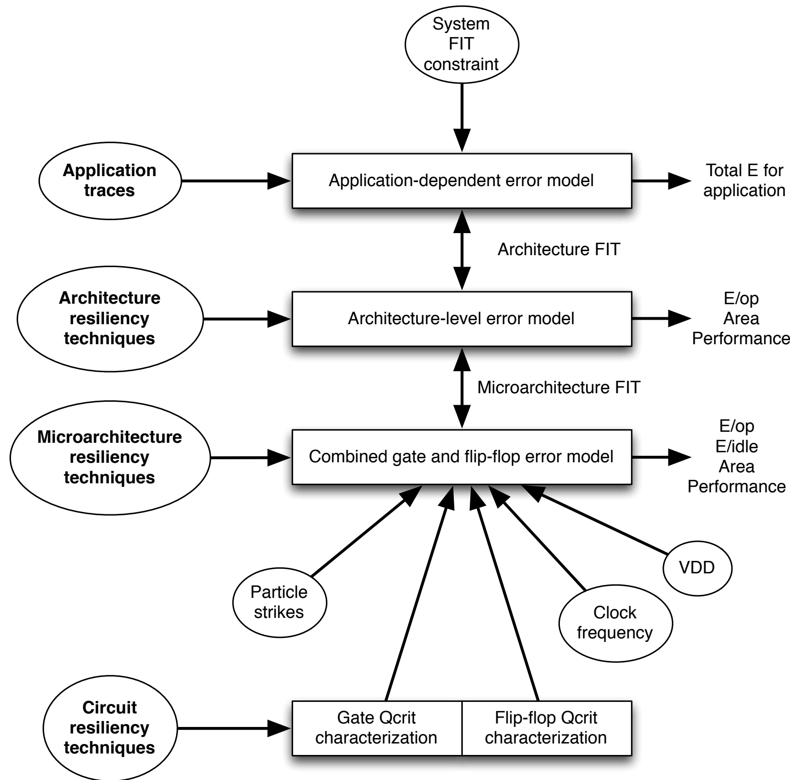

Figure 5.1: Logic resiliency model for transient errors.

5.1.1 Flip-flops

The failure rate of a flip-flop is largely independent of the surrounding circuit, assuming every stored value is significant. Flops are sequential elements, so their outputs depend on both the inputs and the stored state. Thus flip-flops are treated almost exactly the same as gates. The following properties were simulated and precalculated for each sequential element in the technology.

1. Sensitive diffusion area for each storage node as a function of input vector and stored state
2. Input capacitance
3. Q_{crit} for each storage node as a function of input vector and stored state

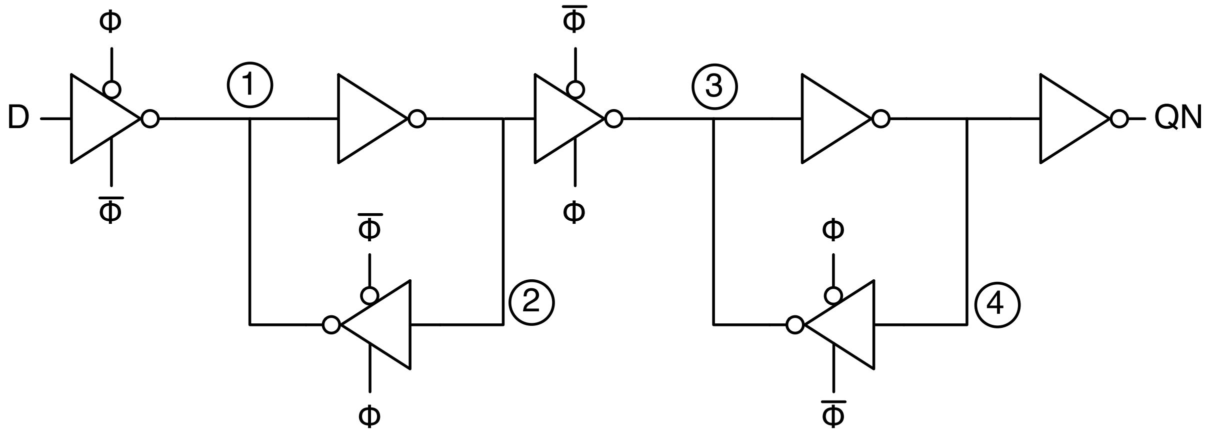

Figure 5.2: C^2MOS D flip-flop with inverted output.

Each sequential element is assumed to contain an output driver, so Q_{crit} is independent of load capacitance. Figure 5.2 shows a typical flip-flop with storage nodes numbered 1-4. The characterization automatically identifies storage nodes of sequential elements, so it also works with latches and flip-flops with other inputs (reset, scan, etc.).

5.1.2 Logic Gates

While the failure rate of a gate depends heavily on the microarchitecture surrounding it, some simplifying assumptions, like ignoring electrical masking, allow gates to be precharacterized for a number of conditions, abstracting away device-level information and speeding up FIT calculations. The following properties were simulated and precalculated for each logic gate in the technology.

1. Sensitive diffusion area as a function of input vector
2. Gate delay as a function of input pin, load capacitance, and voltage
3. Input capacitance

Complex logic gates may have internal nodes sensitive to radiation, such as the node in an AND gate between the NAND gate and the inverter. These complex gates were

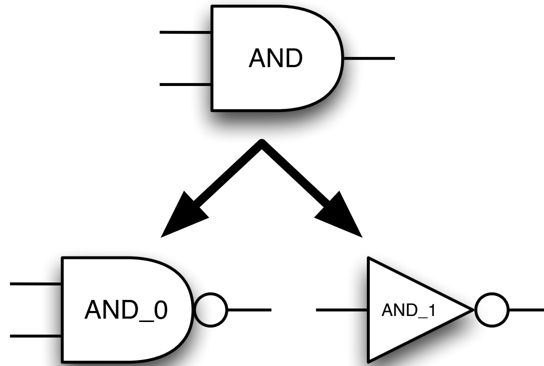

Figure 5.3: Complex gates were split into primitives, and each primitive was separately precharacterized.

automatically split into their primitive gates as shown in Figure 5.3, and each primitive gate was precharacterized.

5.2 Microarchitecture-Level Error Models

5.2.1 Flip-flops

At the microarchitecture level, flip-flops contribute directly to the failure rate. If a particle strikes the storage node of a flip-flop with enough charge, the storage node may flip. There are three assumptions made here.

1. The particle must impart enough charge to flip the stored value. The critical charge at which this happens is precalculated through SPICE-level simulation (see Section 5.1).
2. Transitions between states represent a small fraction of the total time, and are thus ignored. Each latch in a flip-flop is either holding or transparent.
3. If a flip-flop stores the wrong value at any time during a cycle, it is an error at the microarchitecture level.

The imparted charge and other particle strike properties are described in Section 2.2. Soft error rates for flip-flops can be calculated from particle properties, input values, flip-flop critical charge values, and the clock frequency. This is done in conjunction with gate FIT rate calculations, and summed over all flip-flops.

5.2.2 Logic Gates

We tested electrical masking for an arbitrary gate as the glitch propagates through a variable number of inverters, and we found that the glitch attenuation is negligible for up to 32 inverters for our technology. Figure 5.4 shows that the critical charge amount necessary for the flip-flop to latch the glitch does not depend on path length for the same load capacitance. However, we also tested varying the capacitance along a path for a given path length. This produced about a 10% maximum difference in Q_{crit}. This suggests we can conservatively

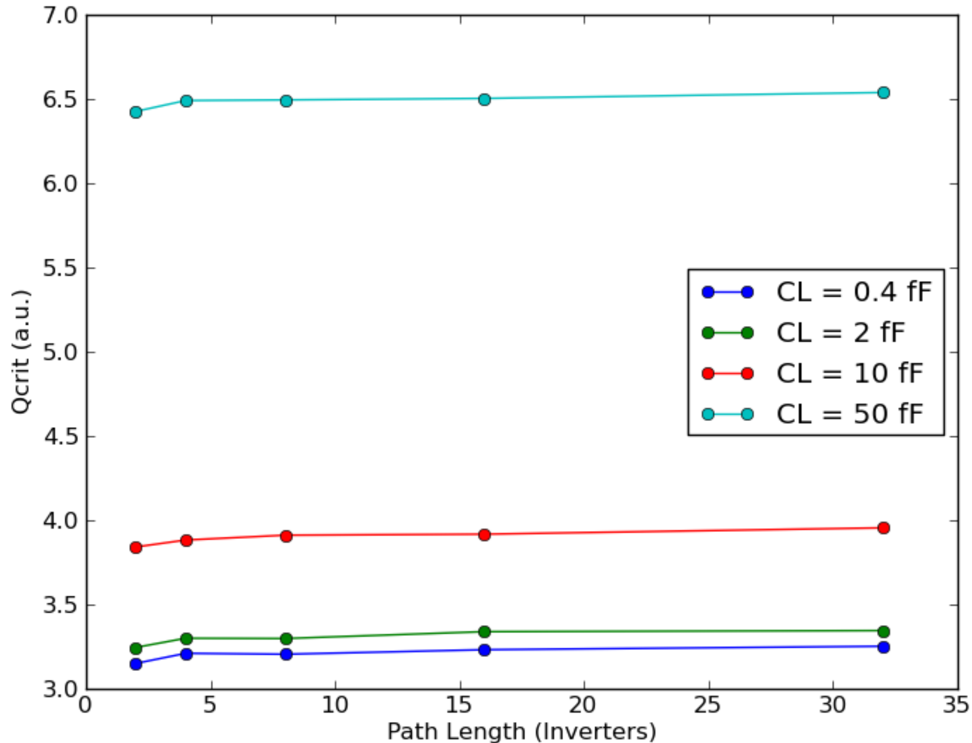

Figure 5.4: $Q_{crit}(t)$ for different path lengths; CL = load capacitance on the gate; struck gate = minimum NAND2 with input 01.

ignore electrical attenuation, simplifying the error calculation. Logical masking is handled by simulating a large number of random input test vectors. Latching window masking is handled by back-annotating gate delays and calculating the probability that a strike with sufficient magnitude occurs at that specific time within the clock period. See the Berkeley FIT estimation tool (BFIT) user's guide for more on calculating the soft error rate of combinational logic gates.[1]

These error calculations are performed on post-synthesis Verilog. Synopsys Design Compiler is used to synthesize the chosen design. Each gate's and flip-flop's contribution to the total FIT is calculated through random input vector simulation with BFIT, and the sum of each gives the total microarchitecture FIT. Area, performance, and idle power estimates are obtained from Synopsys Design Compiler reports. Synopsys VCS is used to simulate random input vectors and obtain nodal activity factors in an SAIF file. Finally, Synopsys PrimeTime reads the SAIF data and reports power, from which energy per operation is calculated.

5.3 Architecture-Level Error Models

At this point, we know which flip-flops are more susceptible than others, but a flip-flop isn't always storing "interesting" data. Combinational logic, such as arithmetic and DSP blocks, may inherently tolerate erroneous data gracefully. For example, a bit flip in the LSB may go unnoticed, especially if the LSB or word is truncated downstream. Sequential logic, such as state machines, contains bits that matter more often. A controller telling the ALU to add instead of subtract will cause a major problem. We assign each flip-flop a weight that reflects how often we care about its value. These weights are conservative, and they simply reflect what percent of time the flip-flop's value, if changed, would affect the output of the simulated benchmark. Multiplying a flip-flop's weight by its failure rate gives a more accurate system-level error rate. Note at this point, the system-level error rate can be made mostly application-agnostic by averaging flip-flop weights from many different benchmarks.

[1] http://www.eecs.berkeley.edu/~holcomb/BFIT.htm

5.4 Application-Level Error Models

All simulation is done on some set of data, so no results are truly application independent. However, the specific data used can strongly affect system-level FIT results. FIT and other metrics at the lower levels are calculated using random input vectors and data. For more accurate FIT and metric data, specific benchmarks are simulated using Synopsys VCS. Node activity is saved and fed into the FIT estimation tool, which generates a revised FIT for the given application. The FIT estimation tool accepts a VCD file as input after some pre-processing. Since designs are pre-processed such that the output of every flip-flop becomes a top-level input, the input file just needs the top-level input data and values stored in each flip-flop each cycle. Synopsys PrimeTime is also run on the new activity data, giving a more accurate application energy measurement. Energy calculations are discussed in Section 5.7.

Raven3, the processor tested by this model, uses the RISC-V instruction set architecture (ISA), an open-source ISA. A benchmark suite was written for the Rocket core, which is a Berkeley implemention of the RISC-V ISA used in Raven3. This suite includes Dhrystone, double-precision generic matrix multiply (DGEMM), and others.

5.5 Resiliency Techniques Tested

Knowing which flip-flops are more error-prone than others suggests that resiliency techniques can be selectively applied. These hardening techniques include time and spatial redundancy, error detection and correction through parity and encoding, margining, and shielding. These are done at different levels of the system hierarchy. Logic datapaths are hardened through parity checking, residue codes, or cyclic codes. Flip-flops employ redundancy through Razor or shadow latches (temporal redundancy) or secondary flip-flops and majority gates (spatial redundancy), such as BISER. These are defined in Chapter 4. These resiliency techniques require special models, but we in general reuse the existing modeling infrastructure and propagate the results up through the hierarchy as usual.

Microarchitecture-level resiliency techniques include selective gate and flip-flop harden-

ing, BISER [35], Razor [32], TMR, ECC, and simple performance scaling (running at a slower clock frequency). Instead of re-simulating for each technique, benchmark metric values are found for an unhardened design and adjusted to account for the resiliency technique used. Since FIT contribution per gate/flip-flop is known, selective hardening is used to limit overhead and maximize FIT reduction. For example, due to logical masking, gates which are far upstream from a flip-flop do not contribute to FIT as much as downstream gates. Thus downstream gates receive priority hardening.

5.6 FIT Estimation

Failure rates are divided here into combinational FIT (CFIT) and sequential FIT (LFIT, or latch FIT, despite including both flip-flops and latches). To calculate failure rates, a free tool developed at Berkeley called BFIT, or the Berkeley FIT Estimation Tool, was used [36]. The tool was modified to work with our technology, ST's 28nm FDSOI process. BFIT uses the standard cell precharacterizations mentioned in Section 5.1.2 to calculate CFIT. The algorithm is described in the user's guide, with some parts restated here for completeness.

First, a gate-level netlist is fed into BFIT, which collapses it into a levelized directed acyclic graph (DAG). The output of each flip-flop is cut, making a new output and input to the DAG. Thus each path ends with a flip-flop, and the DAG outputs represent the state. Input vectors are fed in and propagated through the graph. Finally, starting at each flip-flop (the clock edge), the algorithm traverses all paths backwards, noting the failure rate of each gate as well as the sensitized delay for each path. This final graph traversal incorporates both latching window masking, by tracking path delays, and logical masking, by traversing only sensitized paths.

5.7 Energy Estimation

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.

This generates an SAIF file, which stores activity factors for each node. For Raven3, specific benchmarks were simulated to get more realistic activity factors. In particular, DGEMM was used in this project. 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 (1000 gates). For certain resiliency techniques, custom cells were designed, like a C-element and a hardened flip-flop. These designs were characterizes for power, producing a Synopsys-compatible Liberty file.

Chapter 6

Model Implementation

6.1 Precharacterizations

Gates were precharacterized for different input vectors, fanouts, and window times. To verify the need for each characterization, a single gate was tested first. Figure 6.1 shows how the critical charge for a NAND2 gate varies based on input vector. As seen, some input vectors can be more than twice as susceptible at typical supply levels. Next, Figure 6.2 shows how a gate is only sensitive within a certain timing window, based on the downstream flip-flop. Setup represents the amount of time *before* the clock edge that the data must arrive to be latched. A flip-flop's setup time sets the *early* arrival boundary of the latching window, as shown in the figure. Hold represents the amount of time *after* the clock edge that the data must remain stable to be latched. A flip-flop's hold time sets the *late* arrival boundary of the latching window. Here we use setup and hold to represent the two edges of the latching window. Even for small clock periods, assuming the gate is sensitive for the entire cycle is very pessimistic, and can result in failure rates two to ten times higher than the actual rate.

The range of fanouts used was chosen from typical values present in Raven3 and the standard cell Liberty NLDM characterizations. Larger fanouts increase the window of sensitivity since the collected charge takes longer to dissipate. But the larger capacitance also increases the critical charge required to cause a soft error. Here fanout and load capacitance are syn-

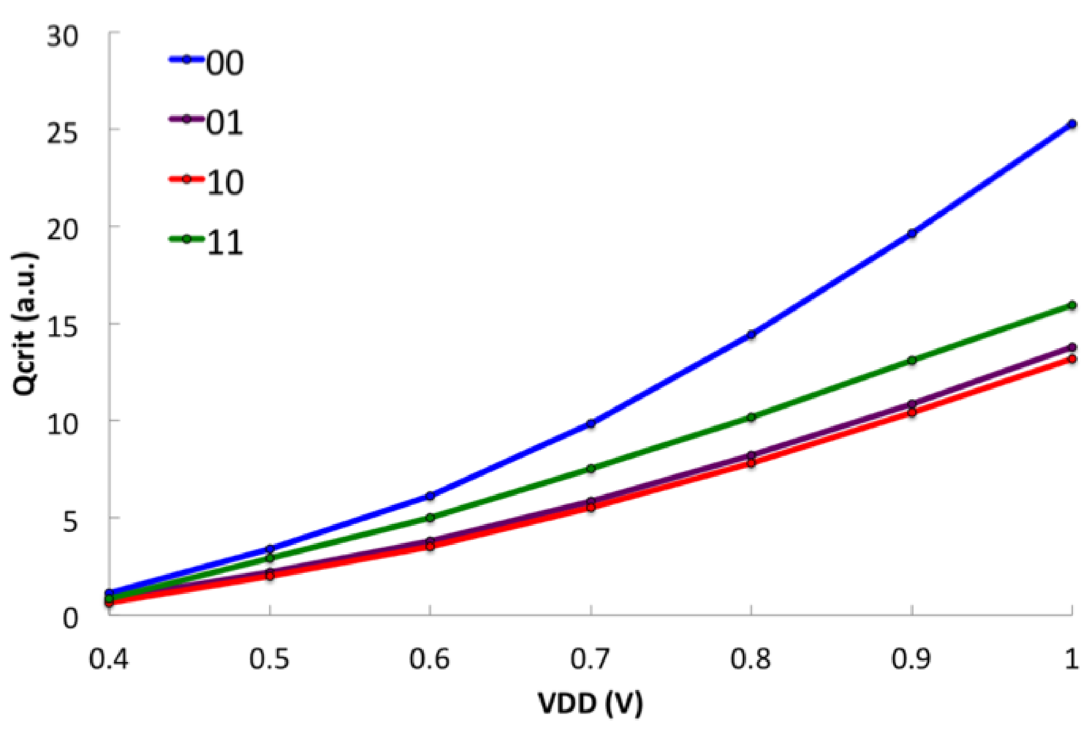

Figure 6.1: NAND2 Q_{crit} depends on the input vector.

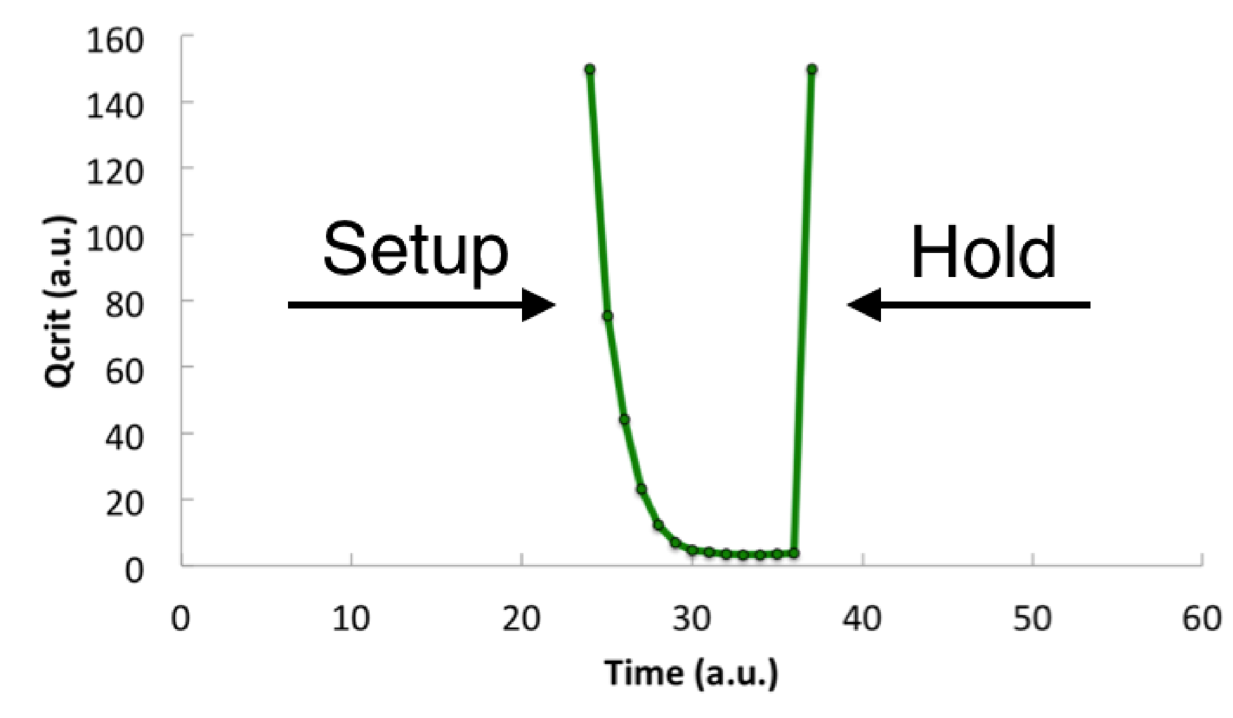

Figure 6.2: Gates are only sensitive within a certain timing window.

onymous, as higher fanout represents a higher load capacitance. For a typical NAND2 gate, the dependence of Q_{crit} on load capacitance is shown in Figure 6.3. As seen, Q_{crit} can change by a factor of 2 or more when considering fanout load. For the same gate, the dependence of the timing window on load capacitance is shown in Figure 6.4, which shows the expected increase in sensitivity window for higher fanouts. Both use 01 as the input vector. Note that these trends are in conflict when considering failure rate. A gate with a higher fanout requires a higher energy particle to cause a soft error, but the gate is sensitive for a longer time within the clock cycle. Later results will show that the increased Q_{crit} affects the FIT more than the increased sensitivity window.

The precharacterizations use a single flip-flop for Q_{crit} and timing window, as characterizing each gate for all possible flip-flops is unreasonable. To estimate the error in choosing a flip-flop, a NAND2 gate was characterized for flip-flops with different drive strengths and features (scan, reset, etc.). Since drive strength is determined by the flip-flop's output driver, this made little difference when characterizing a gate. Figure 6.5 shows that Q_{crit} values can vary by 10% depending on which flip-flop is used. The variation is likely caused by different input and master latch stages. Enable signals, reset signals, and scan signals all add transistors to the input and/or master latch stage of a flip-flop. While Q_{crit} varies only by 10%, the chosen flip-flops can produce timing window variations of 30%, as shown in Figure 6.6. However, it's important to note that the flip-flops with small timing windows were latches, and not registers, so the timing margin variation among registers is smaller. Sequential elements in Raven3 are primarily registers, with latches used only for clock gating elements. No latches were used in the ISCAS benchmarks, since clock gating was turned off.

CFIT excludes sequential elements, whose contributions (LFIT) can be significantly higher than that of logic elements. To calculate LFIT, we also precharacterized flip-flops, as described earlier. For each storage node, the sensitive area and Q_{crit} values were calculated for every input vector and every possible state. During simulation, LFIT is calculated for each flip-flop using its state and input vector, then summed over all the flip-flops.

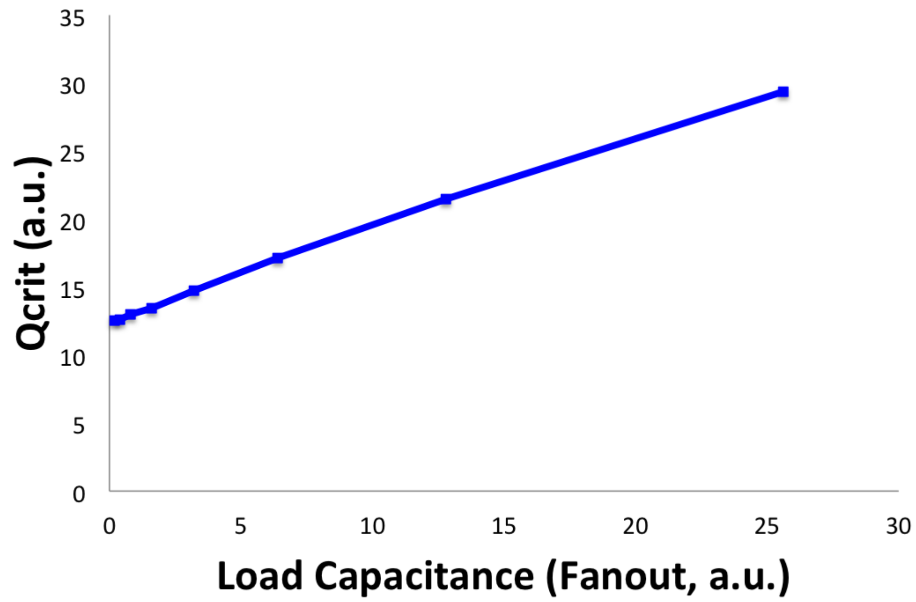

Figure 6.3: NAND2 Q_{crit} depends on the load capacitance.

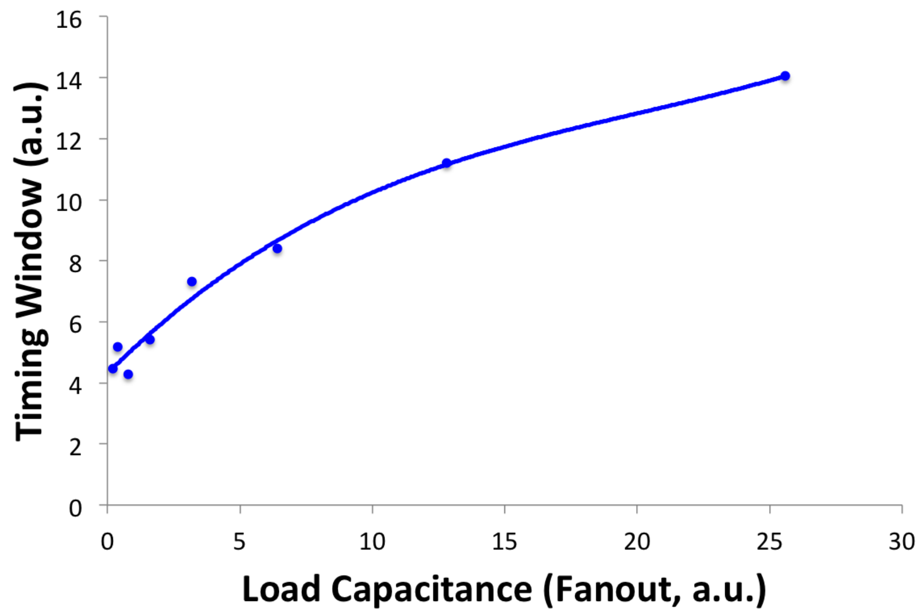

Figure 6.4: NAND2 timing window depends on the load capacitance.

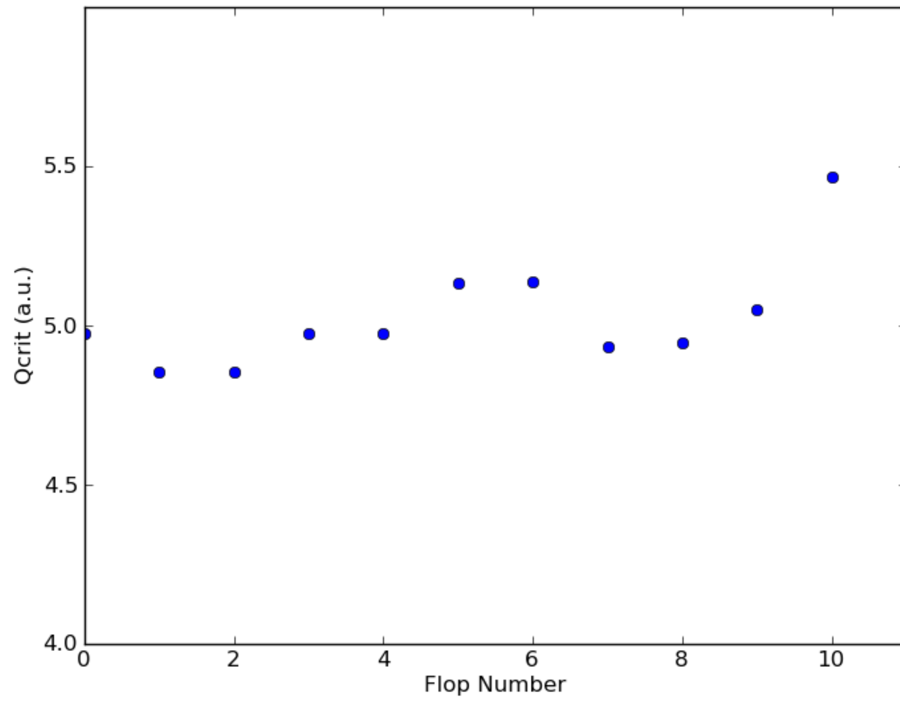

Figure 6.5: NAND2 Q_{crit} depends on the flip-flop used for characterization.

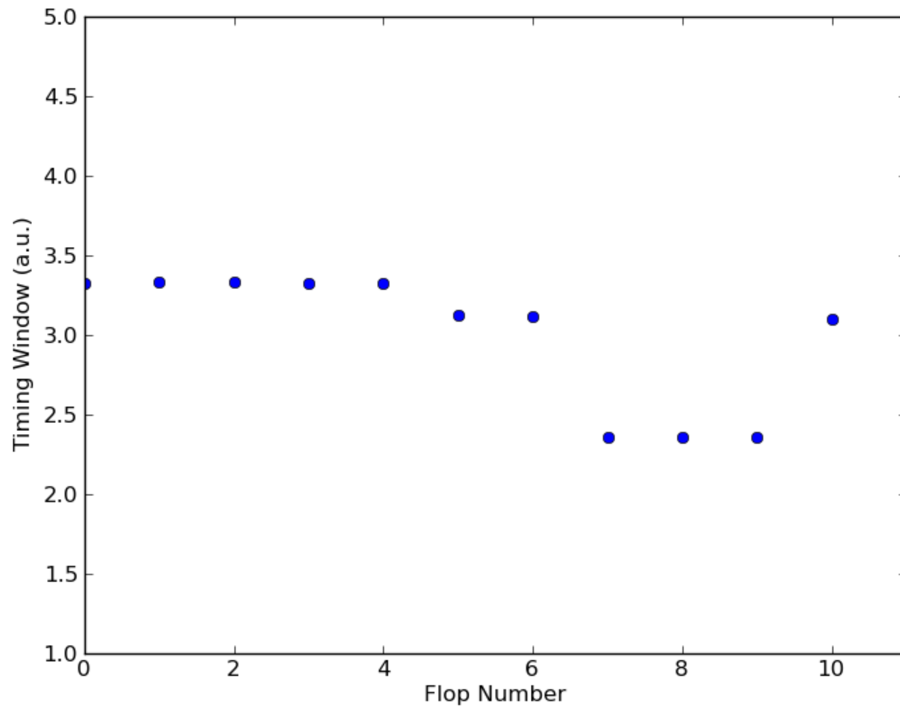

Figure 6.6: NAND2 timing window depends on the flip-flop used for characterization.

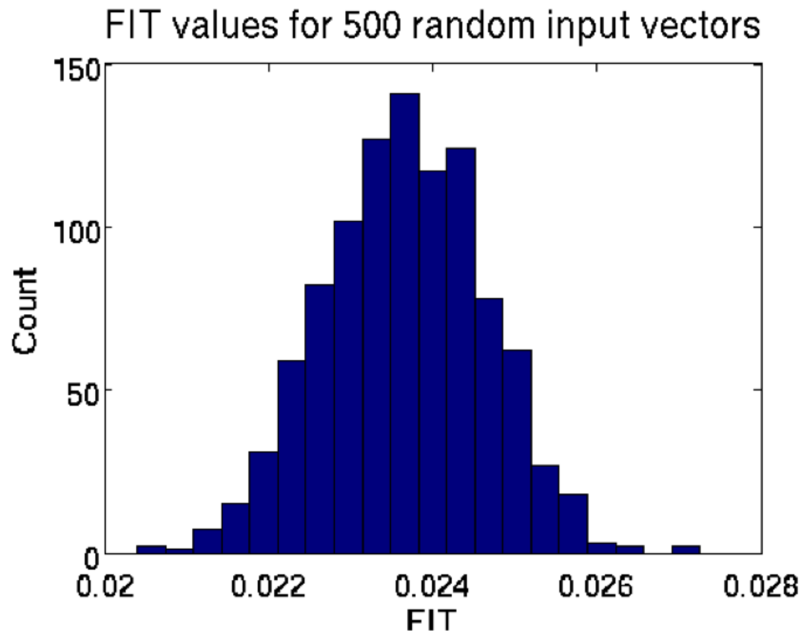

Figure 6.7: Distribution of FIT rates for Raven3 at 1 V and 1 GHz with 500 random test vectors.

6.2 Number of Test Vectors

Since simulating a large number of random test vectors within BFIT is computationally expensive, it makes sense to statistically determine the required number of test vectors. First, a large number of random test vectors were simulated for Raven3 at 1 V and 1 GHz. The results, given in Figure 6.7, show Gaussian-shaped bell curve with a mean FIT of 0.0237 and a standard deviation of 9.76×10^{-4}. Assuming a normal distribution, these data suggest that with only 100 test vectors, we are 99% confident that the FIT result is within 1% of the true value. Thus 100 test vectors were used for all Raven3 FIT simulations.

6.3 Results on ISCAS Benchmarks

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

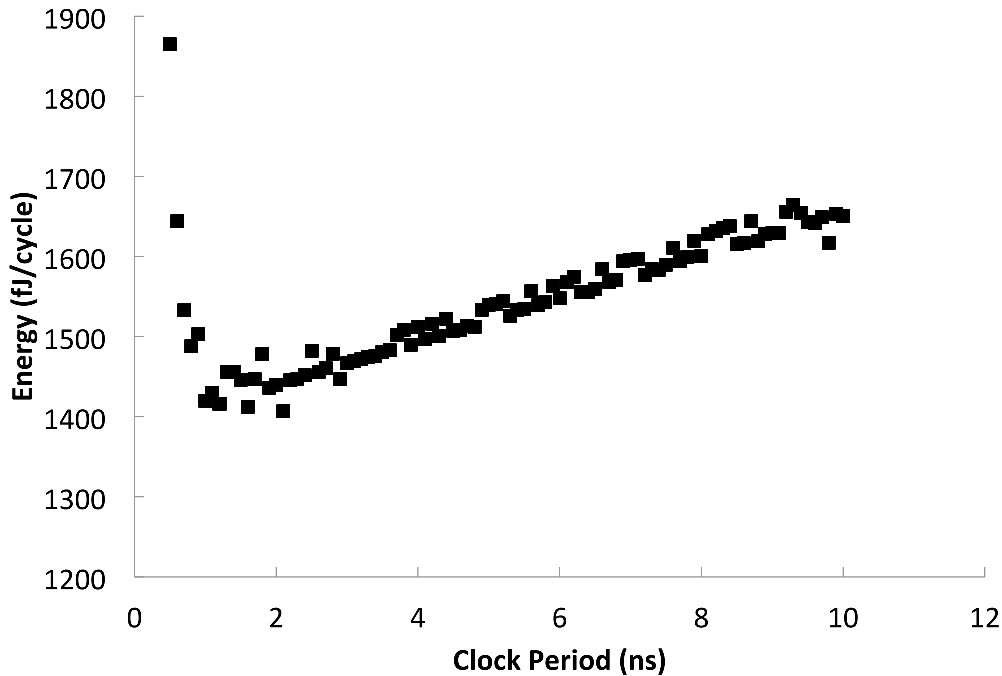

Figure 6.8: Energy vs. clock period for the s1238 benchmark.

6.3.1 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 6.8 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 flip-flop. This trend is evinced in Figure 6.9. We neglected LFIT for this calculation since flip-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 6.10 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, which occurs at 2ns for the s1238 benchmark. 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.

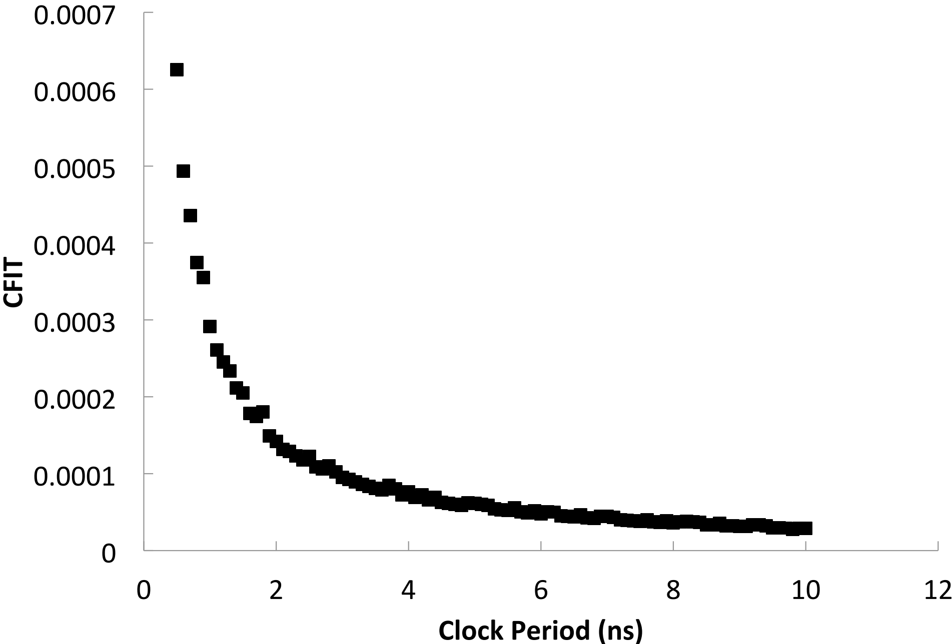

Figure 6.9: FIT vs. clock period for the s1238 benchmark.

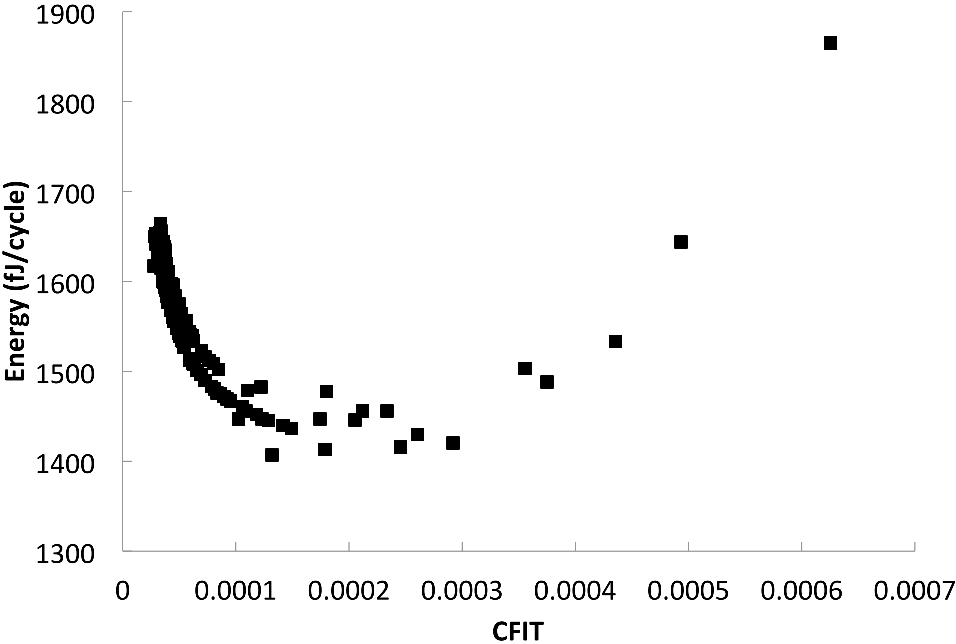

Figure 6.10: Energy vs. FIT for the s1238 benchmark as clock period changes.

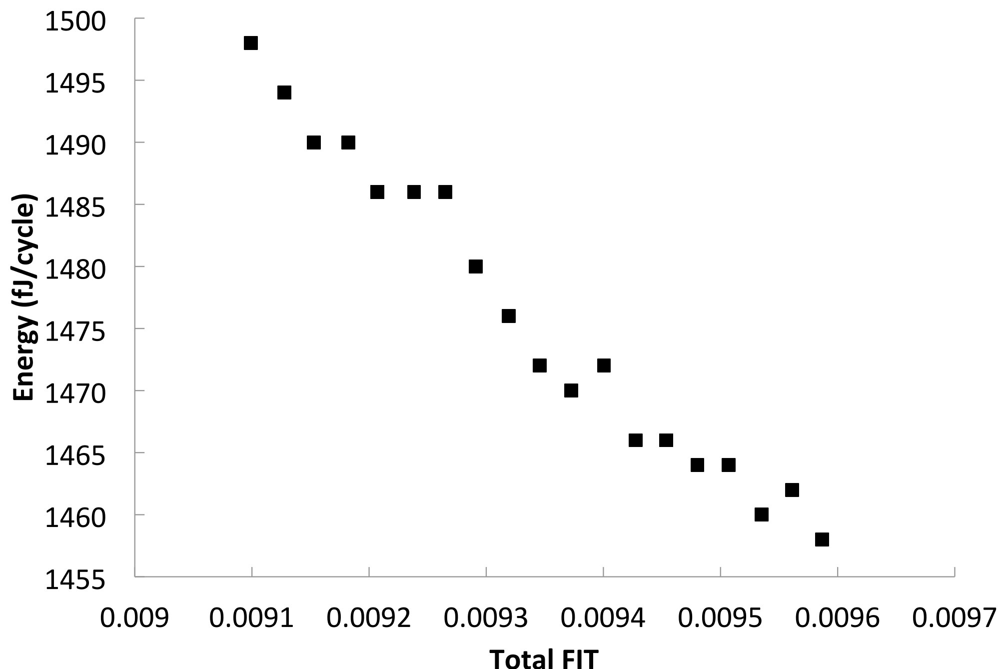

Figure 6.11: Energy vs. FIT for the s1238 benchmark as percent registers hardened changes. Note the graph is fairly flat; the energy varies little. Also note the leftmost point has all registers hardened, and the rightmost point has no registers hardened.

6.3.2 Register hardening

Registers were hardened by manually creating a register with larger storage node transistors. While this increases capacitance and thus Q_{crit} for the storage nodes, it also increases the sensitive area. However, the net result is still a reduction in FIT as more registers are replaced with their upsized counterpart. With respect to energy, the hardened register has a greater 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 6.11 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.

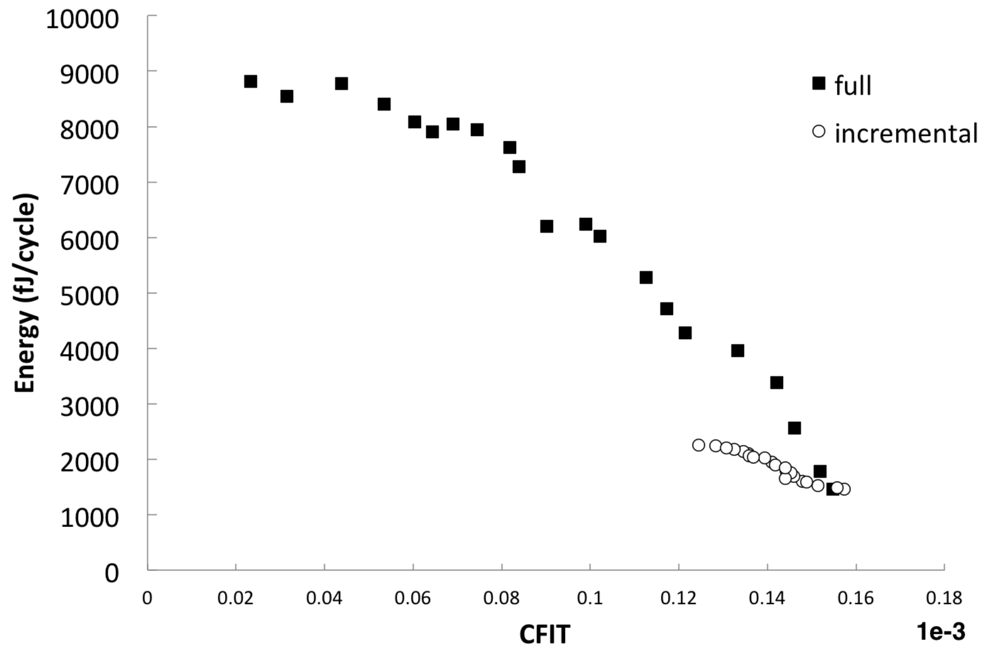

Figure 6.12: Energy vs. FIT for the s1238 benchmark as percent gates hardened changes.

6.3.3 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 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 6.12 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.

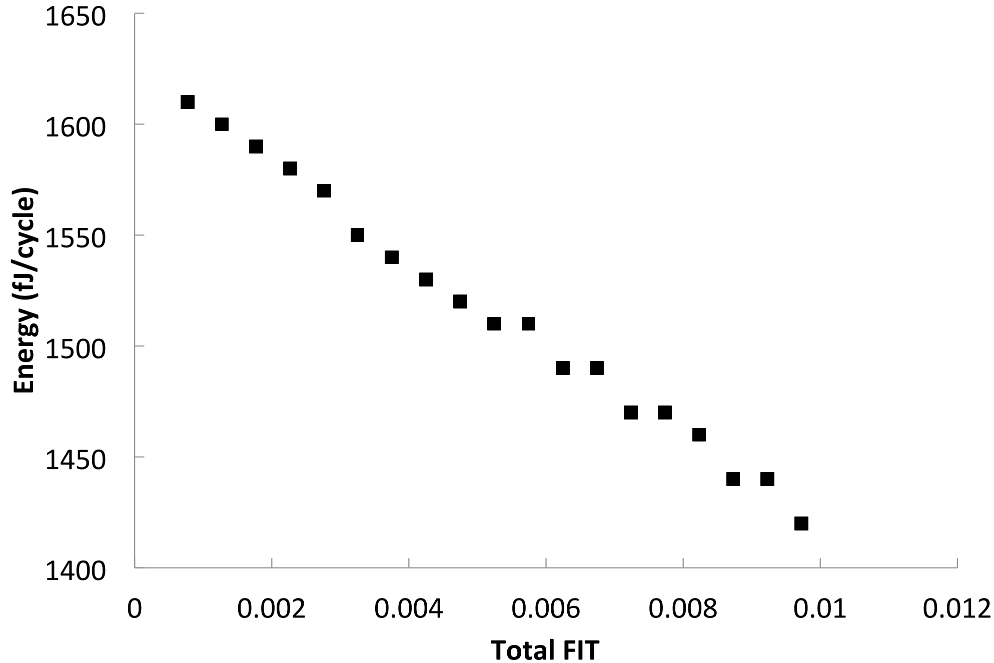

Figure 6.13: Energy vs. FIT for the s1238 benchmark as percent registers hardened with BISER-FF changes.

6.3.4 BISER

As presented in [29], BISER has three versions for register-based designs. The first improves LFIT by 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 [29], we obtained plot in Figure 6.13. 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

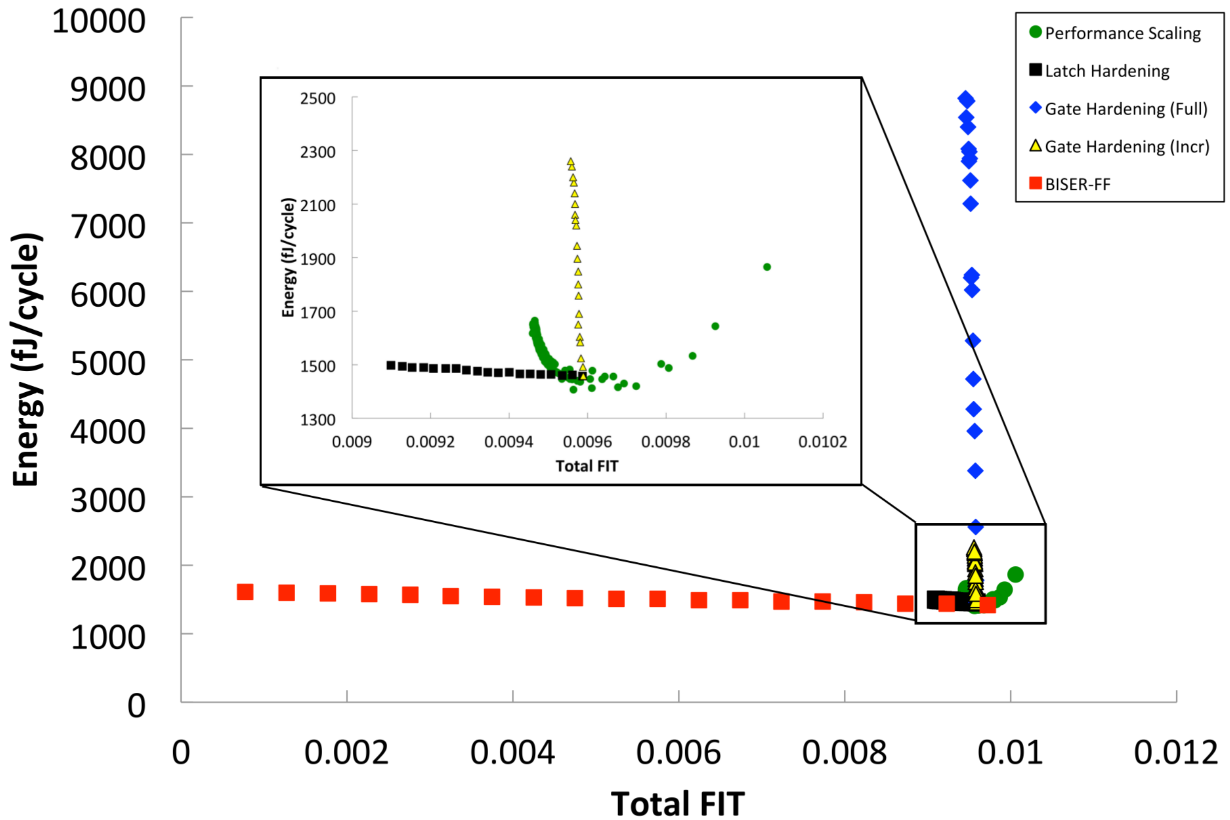

Figure 6.14: Energy vs. FIT for the s1238 benchmark for all resiliency techniques explored.

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.

6.3.5 All resiliency techniques

To properly compare all techniques, we calculated total FIT for each and combined them onto one plot. Figure 6.14 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.

However, a more interesting question is which combinational logic technique is best. Figure 6.15 shows that upsizing gates has a much larger energy overhead than just slowing

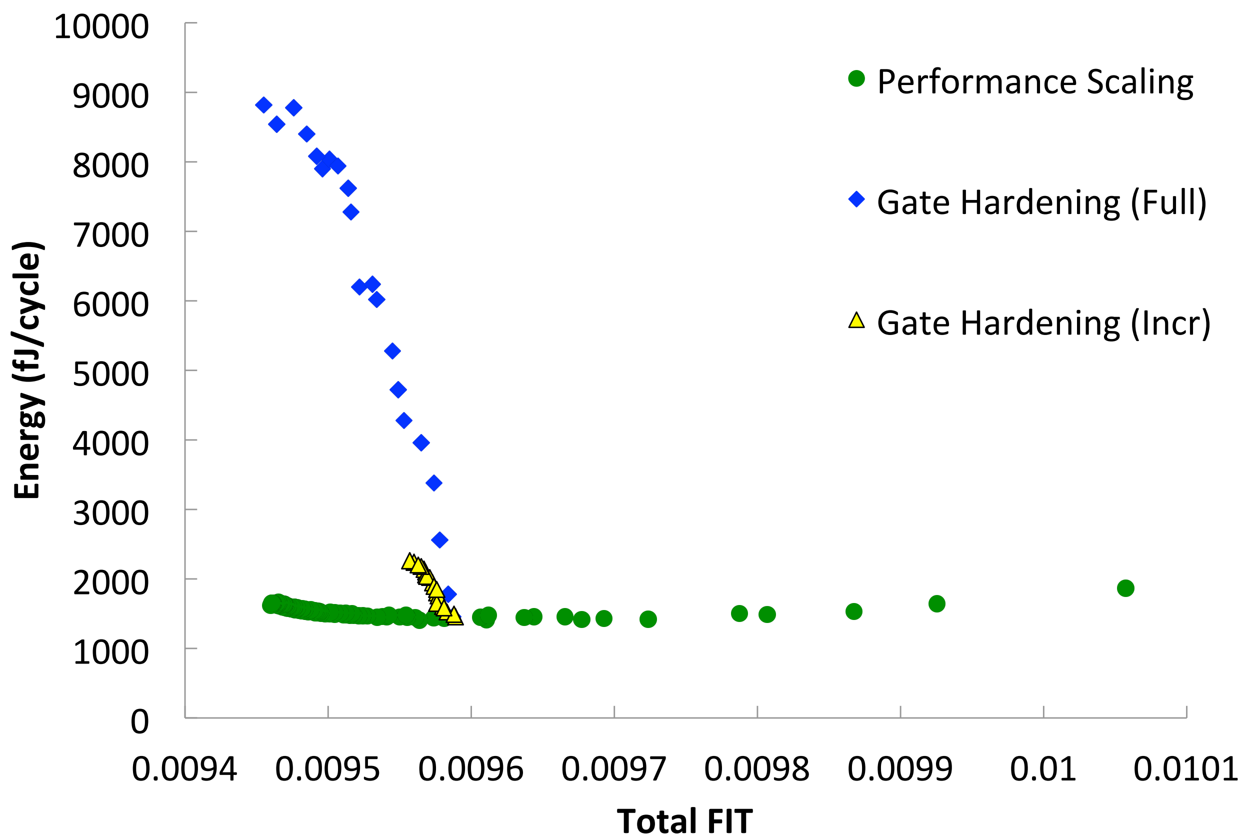

Figure 6.15: Energy vs. FIT for the s1238 benchmark for logic resiliency techniques explored.

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.

6.4 Results on Raven3

The same simulations run on the ISCAS benchmarks were run on Raven3. Raven3 has 390,000 gates after synthesis, which ignores physical-only cells like filler and well taps but includes the level shifters. The results for frequency scaling and gate hardening are shown in Figures 6.17 and 6.16. As seen, by upsizing gates, we trade energy efficiency for failure rate. Also, since Raven3 is designed to run at 1 GHz, the energy efficiency at lower frequencies is worse. At low frequencies, Raven3 spends more time leaking and less time computing. However, these lower frequencies make the particle strike sensitive timing window a smaller

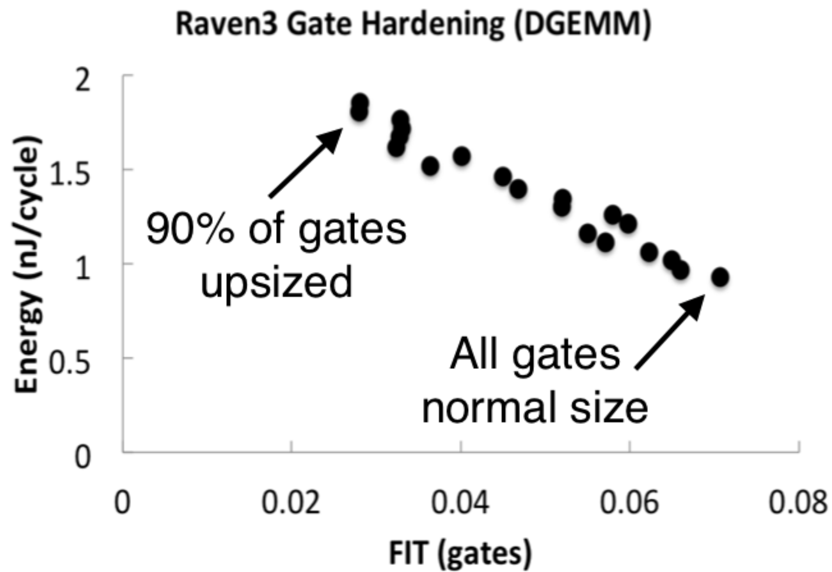

Figure 6.16: Energy vs. FIT for Raven3 as gates are upsized.

fraction of the cycle, reducing the FIT. Thus, just by operating the chip at a slower frequency, we again trade energy efficiency for failure rate.

We separated combinational FIT by functional unit, shown in Figure 6.18. The vector and floating point units comprise most of the logic and most of FIT. These compute blocks should receive the most attention when deciding how to harden the microprocessor.

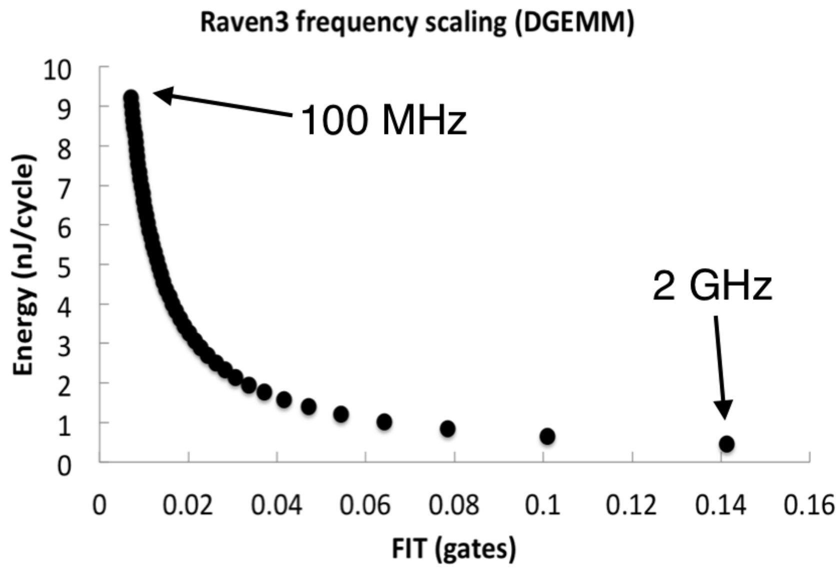

Figure 6.17: Energy vs. FIT for Raven3 as the clock period is adjusted.

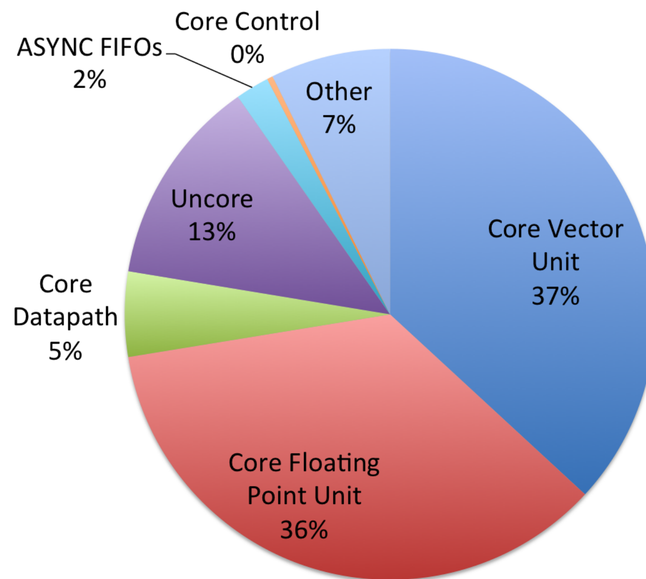

Figure 6.18: Breakdown of combinational logic FIT for Raven3 by functional block. "Other" consists of mostly logic surrounding memories, like the caches and TLB.

Chapter 7

Conclusion

This work presented a summary of existing models for radiation-induced soft errors in generic combinational and sequential logic. Most models begin by approximating a particle strike as a current pulse injected into the circuit. If this pulse disturbs the state during computation, it is called a soft error. The actual failure rate is found by summing the individual soft error rates and derating for factors like timing vulnerability, architectural vulnerability, and application-specific vulnerability. The proposed model takes advantage of some simplifying assumptions to abstract current pulses up to the gate level, which speeds up simulation time. The results suggest that hardening flip-flops is more efficient over hardening gates. Hardening flip-flops can reduce the failure rate by 10x with a 15% energy overhead, while hardening gates can reduce the failure rate by 1% with an 8x energy overhead.

7.1 Key Contributions

This project contributes a framework useful for characterizing a digital ASIC or processor design for radiation hardness. Runtime is reduced through standard cell precharacterization. Resiliency techniques are automatically applied, and trade-offs between energy and resiliency are automatically calculated. The flow works with a realistic technology, ST's 28nm FDSOI. It works on realistic designs, exemplified by the results on Raven3, a low-power mobile

processor with on-chip DVFS taped-out in the 28nm technology in August of 2013. The results of this flow suggest priority be given to hardening flip-flops and memory.

7.2 Future Work

A number of technology-dependent parameters were estimated from trends or publications about similar technologies. A more accurate approach would be to use 3D TCAD or similar simulations on the particular technology used. The following parameters need refinement:

- Q_s, the charge collection parameter in Equation 2.1
- τ_a and τ_b, drift and diffusion time constants in Equation 2.2

A much more accurate simulation approach would use input vectors derived from VCS simulation instead of random values. This would allow the model to accurately predict failure rates for given benchmarks, such as Dhrystone or DGEMM. Also, post-synthesis results are not very realistic, as designs change significantly during place-and-route. A post place-and-route netlist, perhaps with a layout-aware FIT calculation, would give better results.

Silicon results are desired. A test chip will soon be designed and irradiated to verify the model. Also, the model's scope can be easily expanded to consider the macro architecture and the application. Finally, while it's hard to generalize some resiliency techniques to arbitrary circuit designs, specific ones will be added to Raven sometime in the future. For example, ECC will be added to the datapath and to bussed flip-flops. And higher-level FIT propagation and resiliency techniques will be researched. In particular, the tool will calculate architecture vulnerability factors (AVFs) as well as more application-specific results.

Bibliography

[1] N. Seifert, "Radiation-induced soft errors: A chip-level modeling perspective," *Foundations and Trends in Electronic Design Automation*, vol. 4, no. 2-3, pp. 99–221, 2010.

[2] S. Uznanski, G. Gasiot, P. Roche, C. Tavernier, and J. Autran, "Single event upset and multiple cell upset modeling in commercial bulk 65-nm cmos srams and flip-flops," *Nuclear Science, IEEE Transactions on*, vol. 57, no. 4, pp. 1876–1883, Aug 2010.

[3] S. Martinie, S. Uznanski, J.-L. Autran, P. Roche, G. Gasiot, D. Munteanu, S. Sauze, P. Loaiza, G. Warot, and M. Zampaolo, "Alpha-emitter induced soft-errors in cmos 130nm sram: Real-time underground experiment and monte-carlo simulation," in *IC Design and Technology (ICICDT), 2010 IEEE International Conference on*, June 2010, pp. 220–223.

[4] S. Uznanski, G. Gasiot, P. Roche, J. Autran, and V. Ferlet-Cavrois, "Monte-carlo based charge sharing investigations on a bulk 65 nm rhbd flip-flop," *Nuclear Science, IEEE Transactions on*, vol. 57, no. 6, pp. 3267–3272, Dec 2010.

[5] S. Uznanski, G. Gasiot, P. Roche, J. Autran, and L. Dugoujon, "Heavy ion characterization and monte carlo simulation on 32 nm cmos bulk technology," *Nuclear Science, IEEE Transactions on*, vol. 58, no. 6, pp. 2652–2657, Dec 2011.

[6] R. Ramanarayanan, V. Degalahal, R. Krishnan, J. S. Kim, V. Narayanan, Y. Xie, M. J. Irwin, and K. Unlu, "Modeling soft errors at the device and logic levels for combinational circuits," *Dependable and Secure Computing*, pp. 202–216, 2009.

[7] K. Bernstein, "Circuit responses to radiation-induced upsets," *MRS Bulletin*, vol. 28, pp. 126–130, 2 2003.

[8] P. Hazucha and C. Svensson, "Impact of CMOS technology scaling on the atmospheric neutron soft error rate," *TNS*, vol. 47, no. 6, pp. 2586–2594, 2000.

[9] R. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies," *Device and Materials Reliability, IEEE Transactions on*, vol. 5, no. 3, pp. 305–316, Sept 2005.

[10] A. Chavan, E. MacDonald, J. Neff, and E. Bozeman, "Radiation hardened flip-flop design for super and sub threshold voltage operation," *Aerospace Conference*, pp. 1–6, 2011.

[11] L. B. Freeman, "Critical charge calculations for a bipolar sram array," *IBM J. Res. Dev.*, vol. 40, no. 1, pp. 119–129, Jan. 1996.

[12] T. Aton, J. Seitchik, K. Joyner, T. Houston, and H. Shichijo, "Direct Measurement for SOI and Bulk Diodes of Single-Event-Upset Charge Collection from Energetic Ions and Alpha Particles," *IEEE Symposium on VLSI Technology*, pp. 98–99, 1996.

[13] 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," *Dependable Systems and Networks*, pp. 389–398, 2002.

[14] G. Hubert, A. Bougerol, F. Miller, N. Buard, L. Anghel, T. Carriere, F. Wrobel, and R. Gaillard, "Prediction of transient induced by neutron/proton in cmos combinational logic cells," *On-Line Testing Symposium*, 2006.

[15] K. Moen, S. Phillips, E. Wilcox, J. Cressler, H. Nayfeh, A. Sutton, J. Warner, S. Buchner, D. McMorrow, G. Vizkelethy, and P. Dodd, "Evaluating the inuence of various body-contacting schemes on single event transients in 45-nm soi cmos," *Transactions on Nuclear Science*, pp. 3366–3372, 2010.

[16] B. Gill, N. Seifert, and V. Zia, "Comparison of alpha-particle and neutron-induced combinational and sequential logic error rates at the 32nm technology node," *2009 IEEE International Reliability Physics Symposium*, pp. 199–205, Apr. 2009.

[17] M. J. Gadlage, P. H. Eaton, J. M. Benedetto, M. Carts, V. Zhu, and T. L. Turflinger, "Digital Device Error Rate Trends in Advanced CMOS Technologies," *IEEE Transactions on Nuclear Science*, vol. 53, no. 6, pp. 3466–3471, Dec. 2006.

[18] H. T. Nguyen, Y. Yagil, N. Seifert, and M. Reitsma, "Chip-Level Soft Error Estimation Method," *IEEE Transactions on Device and Materials Reliability*, vol. 5, no. 3, pp. 365–381, 2005.

[19] F. Yang and R. Saleh, "Simulation and analysis of transient faults in digital circuits," *Solid-State Circuits, IEEE Journal of*, vol. 27, no. 3, pp. 258–264, Mar 1992.

[20] M. A. Horowitz, "Timing models for mos circuits," Stanford, CA, USA, Tech. Rep., 1983.

[21] N. Seifert and N. Tam, "Timing Vulnerability Factors of Sequentials," *IEEE Transactions on Device and Materials Reliability*, vol. 4, no. 3, pp. 516–522, 2004.

[22] S. S. Mukherjee, C. T. Weaver, J. Emer, S. K. Reinhardt, and T. Austin, "Measuring architectural vulnerability factors," *Micro, IEEE*, vol. 23, no. 6, pp. 70–75, 2003.

[23] S. Kim and A. K. Somani, "Soft error sensitivity characterization for microprocessor dependability enhancement strategy," *Proceedings International Conference on Dependable Systems and Networks*, pp. 416–425, 2002. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1028927

[24] Y. S. Dhillon, A. U. Diril, and A. Chatterjee, "Soft-Error Tolerance Analysis and Optimization of Nanometer Circuits," in *Design, Automation and Test in Europe*, 2005.

[25] H.-h. K. Lee, K. Lilja, M. Bounasser, P. Relangi, I. R. Linscott, U. S. Inan, and S. Mitra, "Design of a Sequential Logic Cell Using LEAP : Layout Design through Error-Aware Transistor Positioning," vol. 94588, pp. 1–6, 2010.

[26] W. Wang and H. Gong, "Edge Triggered Pulse Latch Design With Delayed Latching Edge for Radiation Hardened Application," *IEEE Transactions on Nuclear Science*, vol. 51, no. 6, pp. 3626–3630, 2004.

[27] Y. Sasaki, K. Namba, and H. Ito, "Soft Error Masking Circuit and Latch Using Schmitt Trigger Circuit," *2006 21st IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, pp. 327–335, Oct. 2006.

[28] S. Lin, Y.-B. Kim, and F. Lombardi, "Soft-Error Hardening Designs of Nanoscale CMOS Latches," *2009 27th IEEE VLSI Test Symposium*, pp. 41–46, May 2009.

[29] S. Mitra, N. Seifert, M. Zhang, Q. Shi, and K. S. Kim, "Robust System Design with Built-In Soft-Error Resilience," *IEEE Computer Society*, no. February, pp. 43–52, 2005.

[30] S. Mitra, M. Zhang, N. Seifert, T. Mak, and K. S. Kim, "Soft Error Resilient System Design through Error Correction," pp. 332–337, 2006.

[31] 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," *IEEE Symposium on VLSI Circuits*, pp. 6–7, 2007.

[32] D. E. et al., "Razor: A low-power pipeline based on circuit-level timing speculation," *International Symposium on Microarchitecture*, pp. 7–18, 2003.

[33] D. Blaauw, S. Kalaiselvan, K. Lai, S. Pant, C. Tokunaga, S. Das, and D. Bull, "Razor II : In Situ Error Detection and Correction for PVT and SER Tolerance," *ISSCC*, pp. 400–402, 2008.

[34] M. Fojtik, D. Fick, Y. Kim, N. Pinckney, D. Harris, D. Blaauw, and D. Sylvester, "Bubble Razor: An architecture-independent approach to timing-error detection and correction," *2012 IEEE International Solid-State Circuits Conference*, pp. 488–490, Feb. 2012. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6177103

[35] B. Zhang, W. S. Wang, and M. Orshansky, "Faser: Fast analysis of soft error susceptibility for cell-based designs," *7th International Symposium on Quality Electronic Design*, pp. 760–765, 2006.

[36] D. Holcomb, S. A. Seshia, and W. Li, "Design as You See FIT : System-Level Soft Error Analysis of Sequential Circuits," *DATE*, 2009.