An Accurate Flip-flop Selection Technique for Reducing Logic SER

Eric L. Hill, Mikko H. Lipasti, Kewal K. Saluja
Department of Electrical and Computer Engineering
University of Wisconsin - Madison
{elhill, mikko, saluja}@ece.wisc.edu

Abstract

The combination of continued technology scaling and increased on-chip transistor densities has made vulnerability to radiation induced soft errors a significant design concern. In particular, the effects of these errors on logic nodes are predicted to play an increasingly large role in determining the overall failure rate of future VLSI chips. While a myriad of techniques have been proposed to mitigate the effects of soft errors, system designers must ensure that the application of these solutions does not come at the expense of other design goals. This work presents a heuristic to selectively apply temporal redundancy to flip-flops within a pipelined logic unit, achieving significant reductions in failures associated with soft errors with minimal overhead.

1. Introduction

As computing systems become increasingly ubiquitous, architects strive to create robust systems capable of operating in a wide variety of environments. In addition to meeting performance and power requirements, engineers now have to spend a significant amount of time ensuring their designs also meet reliability goals. The combination of continued technology scaling and increased on-chip transistor densities have made vulnerability to radiation-induced soft errors a significant design concern [8]. The majority of work on soft errors has focused on understanding and mitigating their effects on storage structures [3]. In terms of reducing the overall failure rate, protecting these structures is a sensible starting point, as the majority of transistors on a conventional microprocessor die are dedicated to some form of storage. However, as a consequence of both the increased use of storage protection schemes and technology scaling, the fraction of the system failure rate due to soft errors on logic nodes has been increasing dramatically. A recent study has predicted that for the 50 nm technology generation the contribution of particle strikes on combinational logic nodes to the overall failure rate will be equivalent to that of storage structures [11].

Since architects are concerned with satisfying multiple design goals simultaneously, low cost reliability solutions which provide error tolerance at minimal amounts of overhead are increasingly attractive. This is especially true for many commodity or embedded architectures, where solutions relying on full or near-full system redundancy are not practical. An area overhead of 10% has been suggested as a reasonable target for academic researchers developing new mitigation techniques [1]. In order to satisfy these constraints, the ability to accurately assess which subcomponents of a system contribute most significantly to the overall failure rate is essential.

In this work, a heuristic for placing soft error detectors (time skewed redundant flip-flops in this work) within a logic unit is presented and evaluated. The presented algorithm is unique in that it enables both qualitative and quantitatively accurate conclusions to be drawn about the effect of each decision made. This accuracy enables logic designers to devote resources toward reliability solutions in the most cost-effective manner possible. We evaluate our technique in the context of hardening pipelined functional units, using the results of our heuristics to place the detectors. We find that for the functional units studied in this work, in the best case a greater than 20X reduction in the soft error rate can be achieved at a cost of less than 10% additional area.

The rest of this paper is organized as follows. Section 2 provides background on soft errors in logic, as well as the motivation for this study. A description of the soft error detectors used for this work is provided in Section 3. Sections 4 and 5 outline our heuristic for detector placement, and the infrastructure built for evaluation, respectively. An experimental evaluation of our technique is presented in Section 6. Related work and conclusions from this study are discussed in Sections 7 and 8.

2. Background and Motivation

2.1. Background on Logic Soft Errors

Radiation induced transient faults, or soft errors, typically originate from two sources. Soft errors can be caused by alpha particles present in packaging materials, or by high-energy neutron particles from cosmic rays. The work done in this study focuses on soft errors caused by neutron particles, as recent studies have shown that cosmic radiation is now the primary source[11]. Cosmic ray flux is dependent on altitude, which means that systems that operate at higher levels of the atmosphere have significantly higher SER than those which operate at sea level[11].

Soft errors occur when radiation particles strike sensitive regions of semiconductor devices, injecting charge. Depending on the sizing of the affected transistor and the amount of charge injected, a single event effect may be induced. If the affected device is part of a memory cell (either in a storage array or a latch), this injected charge could potentially flip the current value stored. This type of event is typically referred to as a single event upset (SEU). If the device is part of a combinational logic gate, the charge injected could trigger the generation of a single event transient (SET). An SET is a transient voltage pulse appearing at the output of the affected combinational gate. SETs only result in errors if they propagate to and alter the value captured by a downstream register (a latch or flipflop, depending on the clocking strategy)[14]. The minimum amount of charge needed to induce a single event effect is referred to as the critical charge or Qcrit[11]. This work is specifically targeted at looking at the effects of SETs, as the primary goal is reducing the logic component of the soft error rate.

Soft error rates are generally expressed using the metric of Failures in Time (FIT), which is defined as the number of failures per 10^9 hours [8]. In general, the FIT rate can be calculated in the manner shown in Equation 1.

$$FIT = (RawStrikeRate) * (Derating) \tag{1}$$

The FIT rate of a system is typically computed by multiplying the Raw Strike Rate (which is a function of the component area and altitude-dependent neutron flux) by a derating factor. The derating factor is defined as the probability that a particle strike on a component manifests itself as an error at an output. There exist three well known masking phenomena (logical, electrical, and timing window) which prevent SETs from propagating to and being latched by circuit outputs [11]. Logical masking occurs when a transient waveform is prevented from propagating from an input to the output of a gate because of a controlling value at one of the other inputs. Timing window masking can occur when a SET propagates successfully from a gate to a downstream flipflop, but does not arrive during the interval of time when the flipflop is sampling its data. Electrical masking occurs when the delay of a gate is larger than the duration of the voltage transient at its input, such that the resulting transient appearing at the output of the gate is attenuated in terms of duration and height. Technology scaling has diminished the significance of each of these masking agents, making soft errors in logic a much larger concern [11]. This work is specifically targeted at reducing the derating component of Equation 1.

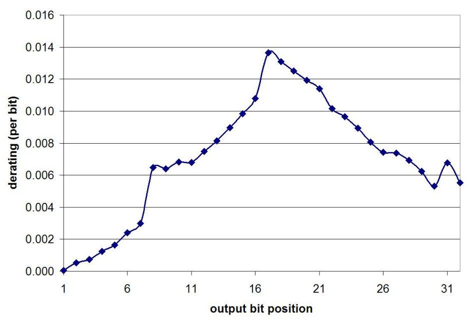

Figure 1. 16x16 multiplier derating per bit.

2.2 Motivation

As discussed previously, SETs only become errors when they propagate from a combinational logic gate to an output and alter the value that is captured by a downstream flipflop. As particle strikes occur with equal probability at any given point in time, individual output bits (flip-flops) in a circuit timing window mask SETs uniformly. In contrast, individual output bits can have differing fan-in cones, meaning that SETs can potentially propagate to individual output bits at varying rates. This essentially means that in contrast to timing window masking, logical masking is not necessarily uniform across output bits. A prior study on estimating SER reports that in multipliers the center bits tend to have an error rate that is orders of magnitude larger than those of the bits closer to the most and least significant positions [12]. The authors of this work refer to this phenomenon as SER peaking [12]. We have also observed this phenomenon by modeling a 16x16 integer multiplier and performing statistical fault injection. Figure 1 shows the amount of errors that occur on each output bit of the multiplier. We believe that this SER peaking phenomena presents an opportunity for low cost soft error protection. Ideally, a combinational multiplier with this behavior could easily be hardened from logic soft errors by simply protecting the subset of output flip-flops where SER peaking occurs.

For combinational circuits, the subset of output flip-flops that need to be protected can be identified by performing statistical fault injection and observing the number of times

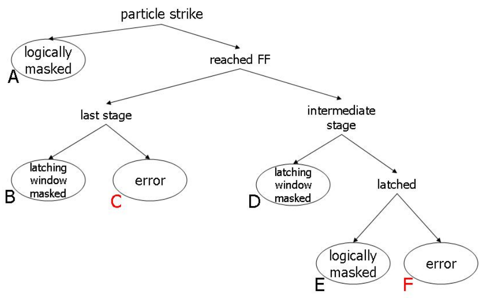

Figure 2. Fault model for strikes on gates in pipelined circuit.

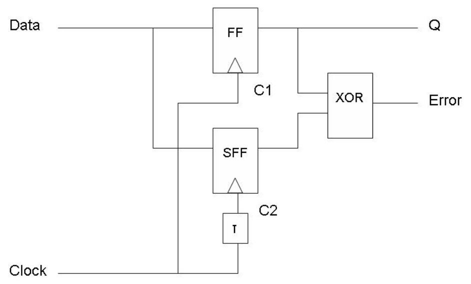

Figure 3. SET detection with time shifted clock.

each output bit is corrupted. Identifying a similar subset of flip-flops in a pipelined circuit is a significantly harder problem. Figure 2 shows our assumed fault model for a SET occurring in a pipelined circuit. This is significantly more complex than the model for a SET in a combinational circuit, which would only consist of outcomes A, B, and C. From this model, it is clear that even if a SET propagates to and is latched by a flip-flop, that error could still potentially be masked as it propagates through the ensuing pipeline stages, never manifesting itself at a circuit output. In addition to this, it is also possible for an SET to corrupt multiple intermediate flip-flops in a circuit, and have only a subset of the corrupted elements be responsible for propagating that error to the outputs. Examples of these scenarios are provided in Section 4. The methodology presented in this work accurately identifies the flip-flops which most significantly impact the failure rate (and thus are the best location to place SET detectors) in the context of this more complex fault model.

3. Overview of SET Detection Techniques

This section of the paper provides an overview of SET detection techniques that could selectively be applied based on the results of our heuristic. We primarily consider solutions which detect SETs by taking multiple flipflop data samples and comparing the values captured. This is accomplished by duplicating the capturing flip-flop, and either time shifting the data input to the duplicate copy, or time shifting the clock input. Both solutions are conceptually similar, but each have unique advantages and disadvantages. This section discusses the trade-offs associated with each option.

3.1. Time Shifted Clock Inputs

This method of SET detection was inspired by the Razor flip-flop proposed by Ernst et al. [6], although it should be noted that the intention of the Razor flip-flop was to allow for more aggressive dynamic voltage scaling instead of improving reliability. The principal idea behind this technique is to duplicate the flip-flop of interest, and supply a delayed clock to one of the flip-flops, as shown in Figure 3. As flip-flops take their input samples on the rising edge of the clock, time shifting the clock inputs allows two different data samples to be taken. A timing diagram of how this technique detects the presence of errors is shown in Figure 4. Both flip-flops take their data samples during the intervals specified by the vertical dotted lines (this work assumes positive edge triggered flip-flops). If there is a mismatch between the two data samples, the pipeline is flushed, and instructions can be re-executed. The main trade-off that must be considered when using a technique like this is related to how much delay is placed between the main and duplicate clock signals. A large amount of skew between the main and shadow clocks detects a large fraction of SETs, but can potentially create short path issues. If the skew between clocks is longer than the shortest path in the circuit, the data sample taken by the shadow flip-flop could be next unit of data propagating through the pipeline, potentially resulting in false positives. The original Razor work dealt with this problem by manually padding short paths [6].

3.2. Time Shifted Data Inputs

Another equivalent method of SET detection, illustrated by Figure 5, is to time shift the data rather than the clock inputs to each flip-flop. This method was inspired by the work presented in [7]. The timing diagram shown in Figure 6 illustrates how this technique can be used to detect errors. Like the previously presented solution, there also exists a trade-off concerning how much skew there should be between the main and shadow data inputs. A large amount of skew can detect a large fraction of SETs, but if the augmented flip-flop is on a critical path, the clock period must be increased to accommodate the skewing delay.

The best technique for a particular logic unit can vary

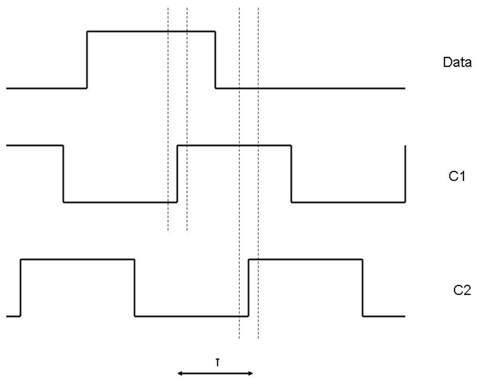

Figure 4. Timing diagram for time shifted clock SET detection.

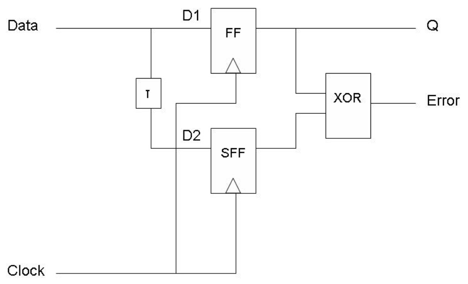

Figure 5. SET detection with time shifted data inputs.

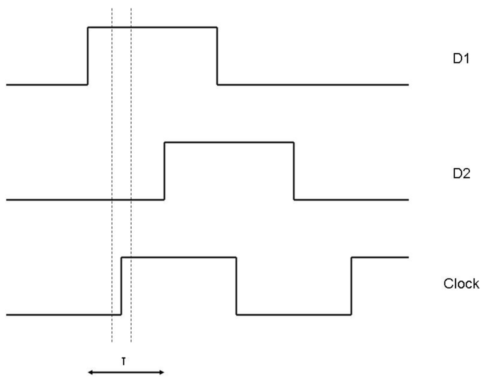

Figure 6. Timing diagram for time shifted data SET detection.

```
1 for (each fault injected)
2   if (error)
3     if (case C)  /* strike in last stage of logic */
4       compute set P /* set of all outputs flipped */
5       score_inc = 1 / cardinality(P)
6       increment counter for each member of P by score_inc
7     else if (case F) /* strike in an intermediate stage */
8       compute set S /* set of flipflops which propagated error */
9       score_inc = 1 / cardinality(S)
10      increment counter for each member of S by score_inc
11
12 sort counters in descending order
```

Figure 7. Selection heuristic pseudo-code.

depending on the characteristics of its timing paths. Time shifting the clock inputs is not an optimal solution for a circuit with a large number of short or zero delay paths as a significant amount of delay padding would be required. In contrast, a circuit with balanced paths could potentially suffer a great deal of delay overhead (in terms of the minimum clock period achievable) if the data inputs were time shifted. For the purposes of this work, all detectors applied used time shifted data inputs, but we believe our results would be applicable for either approach.

4. Flipflop Selection

In this section we present our heuristic for selecting flipflops to augment with soft error detectors. Prior to statistical fault injection, each flipflop in the circuit is allocated a counter. This counter represents the overall contribution (of the corresponding flip-fop) to the circuit failure rate. The pseudo-code for our proposed selection heuristic is shown in Figure 7. Referencing the fault model shown in Figure 2, an error is defined as a particle strike which results in either outcome C (a SET occurring in the last stage of logic and subsequently altering the value of captured by an output flipflop) or outcome F (a SET occurring in an intermediate stage and propagating to an output flipflop).

An example of outcome C (represented by lines 3-6 in Figure 7) is shown in Figure 8. In the last stage, gate A is affected by a SET, which subsequently causes outputs 4 and 5 to capture altered values. Let the set P (referenced in line 4 of Figure 7) represent all of the flip-flops that store an incorrect transient value. Because the transient in this case occurred in the last stage of logic, all flip-flops in the set P must be protected to remove this error. Lines 5-6 in the pseudo-code record the role of each flipflop in causing the error by incrementing the corresponding counters by (1/cardinality(P)). In this case, the counters for flip-flops 4 and 5 are each incremented by 1/2. The reason that each counter is incremented by (1 / cardinality(P)) is to attach extra weight (in terms of error contribution) to cases where a smaller number of flip-flops capture altered values.

Consider the alternative example shown in Figure 9. In this case, gate A is again affected by a SET, but due to different input stimulus, only output 6 in its fanout cone captures an erroneous value. In this example set P only has one member, meaning that the counter for flipflop 6 is incremented by 1. Updating counters in this manner ensures that

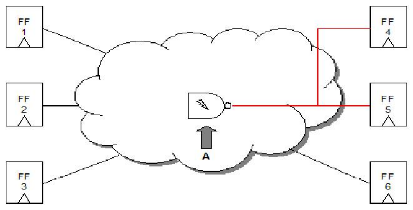

Figure 8. Example of SET in final pipeline stage.

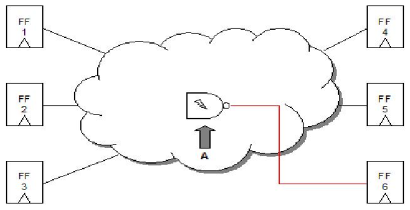

Figure 9. Alternate example of SET in final pipeline stage.

flipflop 6 is credited with having a more significant impact on the overall error rate than flip-flops 4 and 5. If only one flipflop can be protected, flipflop 6 is the correct choice as that would eliminate the error in Figure 9. If either flipflop 4 or 5 was protected, the error in Figure 8 would still exist.

An example of outcome F (represented by lines 7-10 in Figure 7) is shown in Figure 10. In this example, a SET is formed at the output of gate B, resulting in transient values being captured by the flip-flops 1 and 3. The incorrect value stored by flipflop 1 eventually propagates to output flipflop 5. Let the set P again represent all flip-flops that store incorrect values (in this case flipflop 1, 2, and 5), and let set S (referenced in line 8) represent flip-flops that in addition to capturing a transient value, are responsible for propagating incorrect values to circuit outputs. For this injected fault, only flipflop 2 belongs to set S (flipflop 1 does not propagate it's bad value, and flipflop 5 does not capture a transient value). In this case, only the flip-flops in set S (in this case flipflop 1) need to be protected in order to prevent this error. Set S is computed by back propagating from all corrupted outputs. The counter for each flipflop belonging to set S is then incremented appropriately.

Following fault injection each counter contains (for its corresponding flipflop) the overall contribution to the circuit error rate. The value stored by each counter represents the number of times (across the fault injection campaign) a flip-flop is responsible for either directly causing an error by capturing a transient value (output flip-flops in the last stage) or indirectly causing an error by capturing a transient value and logically propagating that value to a circuit output (flip-flops in intermediate stages). A counter with a high value implies that the associated flipflop is more likely to capture and/or logically propagate a transient value, and thus would be an ideal candidate for protection. Sorting these counters (performed on line 12 of Figure 7) creates a list of flip-flops ranked according how much of an overall benefit could be obtained by augmenting a particular sequential element with a soft error detector.

By first normalizing and plotting these counter values, a cumulative density function is created, allowing a logic designer to reason about the theoretical maximum of soft error protection that can be achieved by protecting some subset of flip-flops. The ranking technique presented in this work is unique in that in addition to identifying which flip-flops are the most likely to capture and propagate transient values, our ranking also gives an accurate quantitative estimate of how much protecting each flipflop impacts the overall error rate. Our heuristic implicitly assumes that the addition of a soft error detector will catch all SETs which propagate to that node. In reality, the fraction of SETs that can be detected at a particular node is dependent on the skew between flipflop data input samples. The effect of this skew parameter and verification of our technique is discussed in more detail in Section 6.

Cumulative density functions (in terms of error coverage) are shown in Figures 11 and 12, corresponding to 4 stage pipelined integer and floating point multipliers, respectively. Error coverage is defined as the number of output errors in the baseline (unprotected) case that are caught by SET detectors. For example, in Figure 11 protecting 20% of the flipflops can yield (in the best case) 60% error coverage. From these figures it is clear that a significantly smaller fraction of the flip-flops are responsible for propagating the majority of error in the floating point unit, as evidenced by the sharper rise in displayed error coverage. This can be attributed to the unit's lack of structural regularity, compared to that of the integer multiplier.

5. Infrastructure

5.1 Simulator

The error modeling framework developed for this study is a combination of circuit and gate level simulation. When performing fault simulation, a trade-off must be made between accuracy and speed. Ideally to get the most accurate results, simulation must be done at the lowest level of

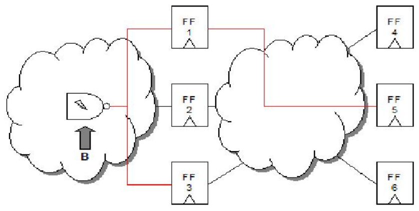

Figure 10. Example of SET in intermediate pipeline stage.

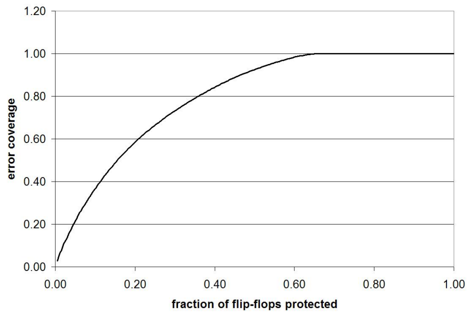

Figure 11. Multiplier Error Coverage CDF.

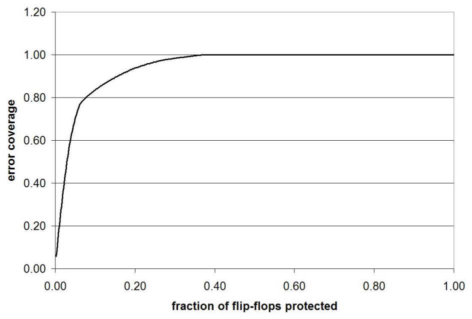

Figure 12. Floating Point Multiplier Error Coverage CDF.

abstraction, using tools like SPICE. Unfortunately, tools at this level are too slow to be able to evaluate a circuit of a practical size. At the other extreme, while cycle accurate simulators typically used in architectural studies can be used to model performance of large systems, and even relative comparisons regarding the vulnerability of storage structures [8], details about gate level structure are often not available at this level of abstraction.

With these concerns in mind, our simulation infrastructure consists of two parts. In order to obtain information

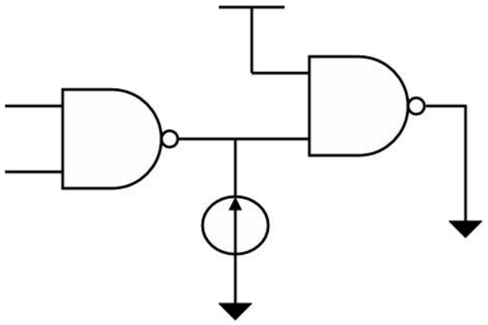

Figure 13. NAND structure used for SET waveform characterization.

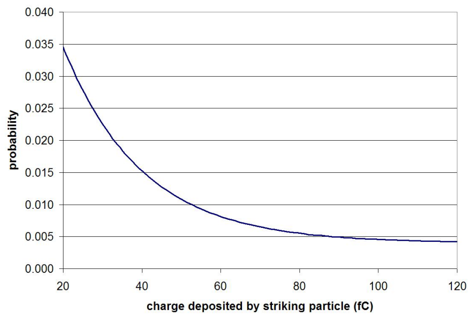

Figure 14. Charge deposition PDF. From [5].

about the waveform characteristics of SETs, particle strikes on combinational logic gates were simulated in SPICE using the 65 nm predictive technology model [2]. Strike events were simulated by first modeling two gates in sequence, injecting a pulse of current at the internal node connecting the the gates and observing the voltage transient at the output, as described in [5]. Specifically we were interested in the duration of the resulting SET, or the amount of time the output voltage was above Vdd/2. An illustration of this setup is shown in Figure 13. The shape of the injected pulse was modeled by a time-dependent exponential function, as described by [11]. The function used is shown in Equation 2.

$$I(t) = Q/T * \sqrt{t/T} * \exp(-t/T) \tag{2}$$

Q and T refer to the amount of charge deposited, and the time constant for charge collection, respectively. The primary goal for this portion of the infrastructure was to map the charge deposition probability distribution function given in [5] to a second function which quantifies the relationship between SET duration and charge deposition. Figure 14 and Figure 15 show the charge probability density and SET duration functions, respectively. These functions are used to drive the gate level component of our simulation frame-

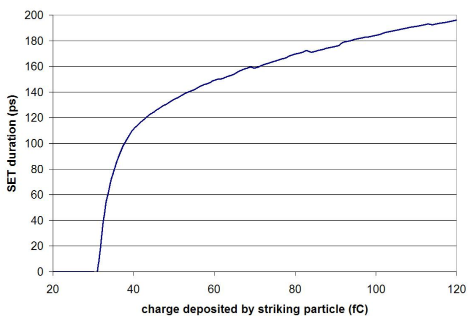

Figure 15. SET duration mapping function.

work.

The second component of our infrastructures is an event driven gate-level simulator. This simulator uses 7-valued logic to simulate the propagation of transients through pipelined circuits. The simulator keeps track of the the events shown in our assumed fault model in Figure 2. Derating can be calculated by dividing the sum of C and F events by the total number of faults injected. It should be noted that our simulator has the ability to accurately model both logical and timing window, but not electrical masking effects. We chose not to model electrical masking effects in this work because the amount of electrical masking that occurs is primarily dependent on the delays of individual gates. This implies that the effects would be similar in both our baseline and protected simulations and would be canceled out when a relative comparison is performed. For each particle strike simulated, a gate, charge deposition value, and an offset into the clock cycle is chosen. The time during the clock cycle is randomly chosen, and the gate and charge deposition values are chosen based on the gate area estimates, and charge deposition probability density function, respectively.

5.2 Benchmarks

In order to create benchmark circuits for evaluation, Verilog behavioral representations of different functional units were first synthesized to elementary logic gates using Synopsis Design Compiler. The resulting net lists were then converted to the ATPG net list format used by our gate level simulator. Area estimates were taken from the corresponding standard cells in the LSI Logic gflxp 0.11um library. These area estimates were used during fault injection to determine the probability of a particle strike occurring on a particular gate. Ideally this area characterization needs to only be done once, as the relative differences of areas between standard cells should remain constant across technology generations.

6. Evaluation

In this section, we present the results of several experiments designed to explore the degree of error tolerance that can realistically be achieved by placing soft error detectors in the manner described previously. In particular the tradeoff between the number of soft error detectors employed and the time skew used to detect errors within each detector is studied. Additionally, the performed experiments serve to validate the CDFs shown in Figures 11 and 12.

The previously mentioned gate level simulator was used to evaluate an integer and a floating point multiplier unit. Each unit was synthesized from a Verilog behavioral representation and pipelined into 4 stages. We felt that these benchmarks are representative of the type of logic units that would be present in a conventional microprocessor. Characteristics of each benchmark circuit are shown in Table 1.

Benchmark	Gates	Flip-flops	Clock Period (ps)
intmult	1981	205	702
fpmult	8194	378	2460

Table 1. Information on Benchmark Circuits

In each experiment, 100,000 faults were injected into each benchmark circuit. Randomized vectors were used for input stimulus. Experiments were performed using detectors that took data samples skewed by 20, 50, 100, and 200 picoseconds, and protecting 5, 10, 25, 50, and 100 percent of flip-flops in each unit. Flip-flops were assigned detectors according to the ranking produced by CDF creation described in Section 4, but different random seeds were used to drive fault injection. The results of these experiments are plotted in Figures 16 and 17 for the floating point and integer multipliers, respectively. The y axis in each figure represents the achieved error coverage and the x axis represents the fraction of flip-flops in the circuit that are augmented with detection logic. The bold vertical line in each graph represents the threshold of 10% area overhead, which was suggested as a target for academic reliability solutions [1].

Both of these figures show that the achievable amount of error coverage increases progressively as both the number of detectors and time skew within detectors increases. This affords logic designers increased flexibility during the design cycle, as units can be augmented with more detectors with less skew if clock cycle time is important, or conversely fewer detectors with larger skew if area is an issue. The error coverage achievable is also significantly different for each unit studied. Looking at Figure 16 adding detectors to 40% of the flipflops (corresponding to a 10% area increase) results in higher than 95% error coverage (over 20X reduction in the SER). In contrast, for the same amount of overhead in the integer multiplier, there is only 65% error

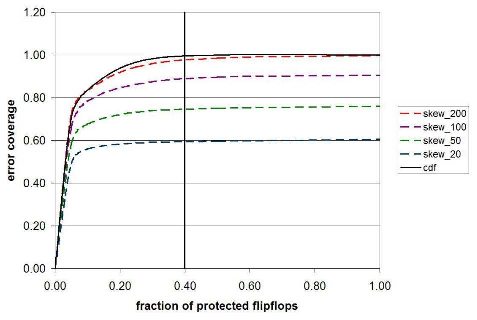

Figure 16. Error coverage vs number of protected flip-flops in floating point multiplier. The vertical line represents a 10% increase in area.

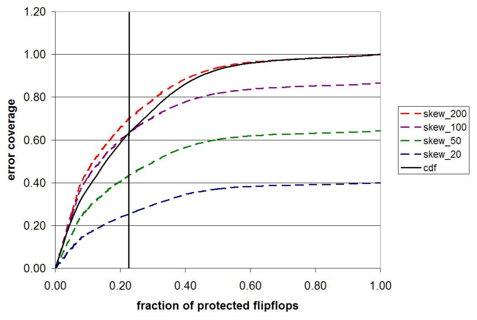

Figure 17. Error coverage vs number of protected flip-flops in integer multiplier. The vertical line represents a 10% increase in area.

coverage (slightly less than 3X reduction the the SER). Depending on the particular unit being considered, our technique alone may be enough to meet reliability goals, or it may need to be complemented with other mitigation techniques. Additionally, designers also must consider the utilization of the functional unit being hardened, which can vary depending on the target applications of the system. The results shown in Figure 16 and 17 also serve to validate the CDF generated by our heuristic. The solid CDF line in both figures tracks closely with the results for 200 ps of skew, which is the maximum transient width used for our fault injection experiments.

7. Related Work

There have been numerous prior works proposing techniques designed to mitigate the effects of soft errors in logic. At the device level, there have been several proposals which reduce vulnerability in logic nodes by increasing the sizes of transistors [15][13]. This raises the critical charge deposited (Qcrit) needed to induce a single event effect, effectively meaning that a larger fraction of particle strikes don't result in errors. As this class of techniques takes a different approach to handling particle strikes, these proposals are complementary to our work. In addition to this, we feel that flipflop based solutions have a significantly greater potential to reduce the number of errors seen in logic. While resizing transistors in a particular CMOS gate will affect particles which strike that gate, augmenting a flipflop with a SET detector can potentially catch any SET that occurs within the fan-in cone of that flipflop.

Another class of techniques, focusing on modifying flip-flops to mitigate soft errors, is more closely related to this work. Rao et al. [9] present a combined approach where both transistors are resized (to increase Qcrit) and flip-flops on paths with timing slack are replaced with flip-flops with larger setup times (to amplify the effects of timing window masking). This work differs from our proposal in that it deals with soft errors by detecting them, rather than increasing the amount of masking. Additionally, approaches that adjust gate or path delays essentially balance all paths in a circuit, resulting in a design that is more susceptible to delay faults which may occur as a result of process variation or wearout.

Blome et al. [4] present a low cost approach to hardening an embedded ARM microprocessor from soft errors through a combination of duplicating a subset of frequently accessed registers, and selectively utilizing time-delayed shadow latches (as soft error detectors). This work also presents a statistical methodology for placing these detectors. This work differs from our work in several ways. First, our work focuses on individual units rather than the entire microprocessor pipeline. We feel that our unit-based approach to hardening components is a good fit for industrial design teams, as multiple analyses can be conducted in parallel. Second, our presented heuristic considers a more detailed fault model, which fully considers the behavior of SETs in pipelined circuits. The use of this model gives our heuristic greater accuracy, and allows for CDF construction with a single simulation pass. Lastly, our study explores trade-offs associated with implementing the SET detectors themselves.

Mitra et al. [10] explore the efficient placement of soft error detectors through the use of formal verification tools. This work is targeted towards control logic, which in general is finite state machine-based. In contrast, our technique is best suited for data path units (like adders and multipliers) and is complementary to this method that deals with control logic.

8. Conclusion

In this work, a novel statistical methodology for reducing logic soft error rates is presented. Our methodology involves selectively adding temporal redundancy to flip-flops within a pipelined circuit to detect soft errors. This work is unique in that the heuristic used for placement takes advantage of the previously studied SER peaking phenomena, and that it is done in the context of pipelined units. Our experimental results show reductions in the logic soft error rate of up to 20X with less than 10% area overhead.

9. Acknowledgments

This work was supported in part by NSF Grant CCF-0702272 and donations from Intel and IBM.

References

[1] *Design Panel for SELSE Workshop 2006*.

[2] *HSPICE PTM – http://www.eas.asu.edu/ ptm*.

[3] A. Biswas, P. Racunas, R. Cheveresan, J. S. Emer, S. S. Mukherjee, and R. Rangan. Computing architectural vulnerability factors for address-based structures. In *ISCA*, pages 532–543, 2005.

[4] J. Blome, S. Gupta, S. Feng, S. Mahlke, and D. Bradley. Cost-efficient soft error protection for embedded microprocessors. In *International Conference on Compilers Architecture Synthesis for Embedded Systems*, October 2006.

[5] H. Deogun, D. Sylvester, and D. Blaauw. Gate-level mitigation techniques for neutron-induced soft error rate. In *ACM/IEEE International Symposium on Quality Electronic Design*, March 2005.

[6] D. Ernst, N. S. Kim, S. Das, S. Pant, T. Pham, R. Rao, C. Ziesler, D. Blaauw, T. Austin, and T. Mudge. Razor: A low-power pipeline based on circuit-level timing speculation. In *ACM/IEEE International Symposium on Microarchitecture (MICRO)*, November 2003.

[7] S. Mitra, M. Zhang, N. Seifert, B. Gill, S. Waqas, and K. S. Kim. Combinational logic soft error correction. In *International Test Conference*, November 2006.

[8] S. Mukherjee, J. Emer, and S. Reinhardt. The soft error problem: an architectural perspective. *High-Performance Computer Architecture, 2005. HPCA-11. 11th International Symposium on*, pages 243–247, 12-16 Feb. 2005.

[9] R. Rao, D. Blaauw, and D. Sylvester. Soft error reduction in combinational logic using gate resizing and flip-flop selection. In *Proceedings of the ACM/IEEE International Conference on Computer-Aided Design (ICCAD)*. ACM/IEEE International Conference on Computer-Aided Design (ICCAD), November 2006.

[10] S. A. Seshia, W. Li, and S. Mitra. Verification-guided soft error resilience. In *Proc. Design Automation and Test in Europe (DATE)*, April 2007.

[11] P. Shivakumar, M. Kistler, S. W. Keckler, D. Burger, and L. Alvisi. Modeling the effect of technology trends on the soft error rate of combinational logic. In *DSN '02: Proceedings of the 2002 International Conference on Dependable Systems and Networks*, pages 389–398, Washington, DC, USA, 2002. IEEE Computer Society.

[12] M. Zhang and N. Shanbhag. A soft error rate analysis (sera) methodology. In *International Conference on Computer Aided Design*, November 2004.

[13] M. Zhang and N. Shanbhag. A cmos design style for logic circuit hardening. In *Proc. IEEE International Reliability Physics Symposium*, pages 223–229, April 2005.

[14] M. Zhang and N. Shanbhag. An energy-efficient circuit technique for single event transient noise-tolerance. In *IEEE International Symposium on Circuits and Systems*, pages 636–639, May 2005.

[15] Q. Zhou and K. Mohanram. Cost-effective radiation hardening technique for combinational logic. In *ICCAD '04: Proceedings of the 2004 IEEE/ACM International conference on Computer-aided design*, pages 100–106, Washington, DC, USA, 2004.