Time Redundant Parity for Low-Cost Transient Error Detection

David J. Palframan, Nam Sung Kim, Mikko H. Lipasti
Department of Electrical and Computer Engineering
University of Wisconsin–Madison
palframan@wisc.edu, nskim3@wisc.edu, mikko@engr.wisc.edu

Abstract—**With shrinking transistor sizes and supply voltages, errors in combinational logic due to radiation particle strikes are on the rise. A broad range of applications will soon require protection from this type of error, requiring an effective and inexpensive solution. Many previously proposed logic protection techniques rely on duplicate logic or latches, incurring high overheads. In this paper, we present a technique for transient error detection using parity trees for power and area efficiency. This approach is highly customizable, allowing adjustment of a number of parameters for optimal error coverage and overhead. We present simulation results comparing our scheme to latch duplication, showing on average greater than 55% savings in area and power overhead for the same error coverage. We also demonstrate adding protection to reach a target logic soft error rate, constituting at best a 59X reduction in the error rate with under 2% power and area overhead.**

I. Introduction

As technology progresses, susceptibility to particle-induced "soft" errors has become a key concern, since smaller device sizes and lower supply voltages dramatically decrease the amount of charge stored per node. Soft errors occur when an alpha or neutron particle passes through a semiconductor device, creating electron-hole pairs. If this generated charge is collected by a transistor source or drain, it can interfere with correct circuit operation [1]. Much previous work focuses on protecting storage elements, where particle strikes can cause bit flips known as single-event upsets (SEUs). Memories are particularly susceptible to these errors and are commonly protected using error-correcting codes (ECC). More recent work also proposes methods for extending protection to pipeline storage elements [2], [3], [4]. It is also possible for soft errors to affect combinational logic in what are known as single-event transients (SETs). In these cases, a transient pulse is created and propagates through the circuit, potentially being latched at the output.

SEUs have received the most attention since they are generally more likely to cause errors than SETs. For instance, a transient may not be present during the flip-flop latching window (timing window masking) or may be attenuated as it propagates through logic gates (electrical masking). Due to these masking effects and the higher soft error resilience of older processes, yesterday's designers could overlook transient faults. Recent estimates, however, indicate that the soft error rate (SER) of combinational logic will soon equal that of memory [5]. Therefore, even non-critical consumer devices will need to address the logic soft error problem to guarantee a basic level of operation. Since moderate levels of fault tolerance are acceptable for these applications, while cost (in terms of die area and power consumption) is paramount, the conventional balance between these design objectives has shifted. This new balance, emphasizing reasonable reliability at a low cost, has only recently begun to be explored.

Reducing the cost of many SET mitigation schemes often involves protecting fewer elements, sacrificing error coverage. Some of these elements (e.g. flip-flops or latches) may be more prone to errors and therefore more worthwhile to protect than others. Based on this insight, Hill et al. present a method for flip-flop selection to maximize SET protection [6].

Instead of simply sacrificing the number of protected elements to reduce cost, we propose a SET protection technique that relies on parity-based compaction of circuit outputs for power and area reduction. Our method is orthogonal to Hill's, since both can be applied independently to minimize cost. We note that using parity for concurrent error detection (CED) is not a new idea. Many previously proposed applications are capable of detecting stuck-at-faults [7], [8], and do so by employing complex parity prediction logic. Our approach is quite different, since it targets transient faults and does not rely on expensive prediction circuitry. Our work includes the following significant contributions:

1) a novel low-cost, time redundant technique using parity for SET detection;
2) discussion of methods for avoiding undetected even errors due to parity;
3) analysis comparing our technique to a latch duplication approach in terms of area overhead, power overhead, and error coverage;
4) demonstrated compatibility with Hill's flip-flop selection scheme for tunable error coverage.

The reminder of this paper is organized as follows: Section II explores previous SET protection techniques and motivates our proposal. Section III introduces and discusses our SET detection scheme. Section IV presents results for protection with our method and with latch duplication, comparing these in terms of error coverage, area overhead, and power overhead. Our results show an average of over 55% savings in power and area overhead while maintaining error coverage. Finally, Section V concludes the paper.

978-3-9810801-7-9/DATE11/©2011 EDAA

II. Background and Motivation

This section presents prior proposals for countering the SET problem. We also discuss the tradeoffs between error detection and error correction, and explain time redundancy.

A. Overview of SET Protection Schemes

Methods for SET mitigation can be applied at the process, circuit, or architecture level [1]. Silicon-on-insulator (SOI) is an example of a process technology that can somewhat reduce the logic SER. Circuit-level approaches often involve hardening certain nodes against errors through local duplication [9] or gate resizing [10]. While these techniques reduce the logic SER, they can have limited effectiveness and often require substantial area overheads. Architectural techniques, on the other hand, usually add external protection circuitry, leaving the original circuit mostly unmodified. Such techniques are flexible, have low overhead, and can be tailored to the architectural definition of an error.

Architectural solutions can either be error-detecting or error-correcting. Since processors often include a mechanism for replaying instructions to recover from misspeculations, a method for soft error detection could also employ this framework. Omitting local correction logic can reduce overhead and allows the processor to determine whether a replay is actually necessary. For a duplication-based protection scheme, having two copies of a circuit enables error detection via result comparison, but not correction. For correction, the circuit could be triplicated with the correct output selected by a majority vote. Though this choice between 100% and 200% overhead is extreme, the tradeoff generally holds. For the remainder of this work, we only consider methods that detect errors due to SETs. One might think that reliance on replay for correction could reduce the number of instructions per cycle (IPC), but soft errors are infrequent enough that this difference is insignificant.

Instead of complete duplication, another class of SET detection techniques adds dedicated error-detection logic. For example, this logic could duplicate only part of a circuit [11], or check the validity of the circuit output based on a set of rules governing output characteristics. These characteristics can be predetermined [12] or predicted in real time [8].

B. Time Redundancy

A third type of SET detection technique exploits time redundancy. In this paper, time redundancy does not imply replaying each instruction multiple times, though this is possible. We use the term in a context similar to [13] and [14], where it has a much lower performance impact. In this version, a circuit output is guaranteed to be stable during a certain period, so that any perturbation indicates the presence of a transient error. To create this window of stability, additional setup or hold time constraints are imposed. In addition, some circuitry is required to check for transients during this time.

One often proposed solution involves double sampling. An additional "shadow" latch is added to sample at a different time than the main latch, but during the stable window. If the two samples are taken sufficiently far apart, the shadow latch will capture the correct value, even if the main latch captures an erroneous value. One way to implement this when imposing a hold time constraint is through a technique such as Razor, which was originally intended to detect timing violations due to dynamic voltage scaling [15]. Razor uses a delayed clock to trigger shadow latches.

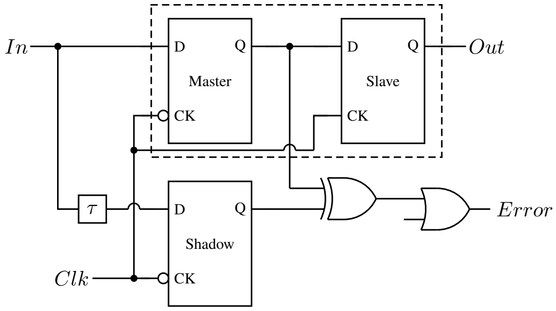


Fig. 1. Flip-flop augmented for SET detection using a shadow latch and delayed data.

The opposite approach is proposed by Mitra et al., in which shadow latches sample early [16]. The same effect can be accomplished in one of two ways: either by using an early clock for shadow latches, or by introducing a data delay, as shown in Figure 1. Here, the XOR gate acts as a comparator to detect discrepancies between the main latch and the shadow latch. This signal is ORed with the outputs of other comparators. Using delayed data has the advantage of not requiring an additional skewed clock. Finally, another possibility is to use two additional latches to sample both before and after the main latch [14].

C. Tradeoffs of Time Redundant Techniques

Regardless of the specific time redundant technique chosen, additional setup or hold constraints must be specifically enforced (if not already met). Doing so may require additional area or cycle time. For instance, a Razor-style technique can require extra logic to pad short paths, ensuring that a new value cannot propagate through the combinational logic in time to be erroneously captured by a shadow latch. Other techniques that rely on post-sampling but not shadow latches have been proposed [17], but the benefit of using compact detector circuitry can be quickly overshadowed by padding overhead as the detection window is lengthened. To ensure additional setup time for the pre-sampling variant, the clock period can be extended. This extra slack determines the allowed skew between samples and therefore the maximum pulse length that can be detected. Slack time can be adjusted by the designer to select a desired level of reliability. Though this approach does slightly degrade performance, we consider it the better option for the multicore era, as reducing power consumption takes precedence over individual core performance.

III. SET Detection using Parity

Parity is an effective data compaction technique for detecting single bit flips in pipeline storage elements. In the simplest pipeline application, a parity tree can be connected to flip-flop inputs, with a duplicate tree connected to flip-flop outputs. On each rising clock edge, parity is captured for the current (old) data as well as the incoming (new) data. Comparing parity of data immediately before it is latched and after it has been held for one or more cycles can detect flip-flop upsets. This check can also be performed with only a single parity tree using a method similar to [3], in which parity trees are connected to flip-flop outputs, and parity bits are latched as soon as possible after flip-flops are updated. These stored parity bits are then continuously compared against current parity to detect flips.

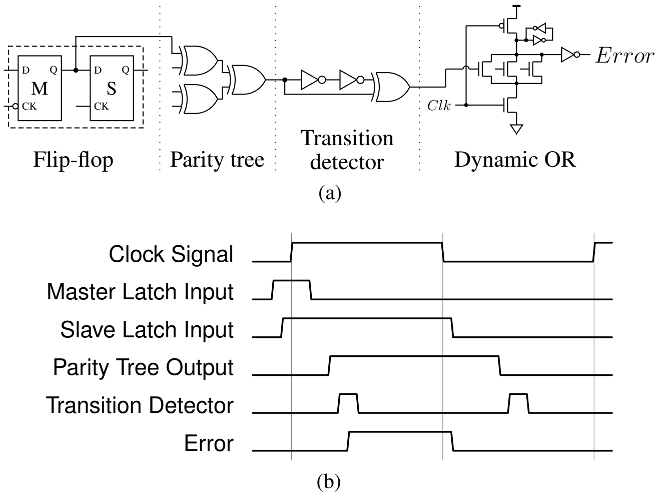
While the low overhead of parity is attractive, it has not been widely considered as a technique for detecting transient faults in combinational logic. The reasoning is simple: SETs are capable of fanning out to corrupt multiple flip-flops. If an even number of bits in a parity group is flipped, the parity bit will retain its original value, and the error may go undetected.

A solution to the even-flip problem is to exploit asymmetric delays in the circuit being protected. Consider the case in which a single strike corrupts multiple flip-flops. For this to occur, the incorrect transient value must be present at the inputs of all corrupted flip-flops during the latching window. Though these pulses overlap during the latching window, they are not necessarily perfectly aligned. With longer pulse widths, it becomes increasingly possible for skewed transients to be captured by multiple flip-flops. A key observation is that logic faults corrupting an even number of flip-flops in a group may still cause the parity bit to fluctuate if the pulses are skewed in time.

A. Detector Circuitry and Basic Operation

Instead of discretely sampling parity bits, we propose the use of a transition detector, such as the one in [18]. Figure 2a illustrates our SET detection setup. As shown, XOR trees are connected between the master and slave latches of each protected flip-flop. This design choice isolates the parity and detector circuits when the clock is high and master latches are opaque. The benefits of this isolation are twofold: 1) assuming that most output glitching occurs during the high clock phase, less power is consumed, and 2) these early transitions are prevented from triggering an error. Parity is monitored by a transition detector, which is connected to a dynamic OR gate. This gate can be used collect the outputs of multiple transition detectors, each having its own parity tree. The error signal from the dynamic logic can be captured by a set-dominant latch, as discussed in [18].

Operation is analogous to the pre-sampling scheme in [16] that uses a padded datapath for shadow latches, requiring only a single clock. Likewise, the parity trees in our scheme provide a delay as well as signal compaction. Our approach also allows the use of a single clock for the main circuit and error detection logic. Figure 2b shows an example scenario in which a transient pulse corrupts a flip-flop and triggers an error. This example shows only one signal connected from a master-slave flip-flop to a parity tree. For simplicity, it is assumed that the correct value of this signal is low, and that all other signals to the parity tree remain low. The transient pulse at the master latch input spans the latching window, meaning that the master latch captures and holds the erroneous value for the first half of the cycle. The signal labeled "Slave Latch Input" is also the input to the parity tree. Though the master latch is opaque when the clock is high, the first edge of the transient has already entered the parity tree before the clock's rising edge. The output of the transition detector is normally low. Because of the delay introduced by the inverter buffer, it produces a brief pulse in response to any rising or falling edge at the input. As the example shows, if this pulse is created while the dynamic OR gate is active (Clk high), the "Error" signal will remain high until the next time its internal node is precharged (Clk low).



(a)

(b)

Fig. 2. (a) Parity-based SET detection circuit; (b) Timing diagram demonstrating SET detection.

Concurrent work by Rossi et al. proposes a parity-based mechanism similar to ours but primarily intended to count occurrences of SETs [19]. This approach connects a parity tree to flip-flop inputs, making transients vulnerable to electrical filtering. For our application, such filtering is not as tolerable. By connecting the parity tree between flip-flop latches, only the first edge of an error-causing transient will propagate through the tree, making our design less vulnerable to this electrical effect.

B. Timing

As previously mentioned, pre-sampling time redundant techniques require additional slack at the end of the clock cycle. Though our example in Figure 2b shares the clock between the flip-flop and the detector dynamic OR, this is not necessarily required. Assuming no errors are masked due to concurrent transitions in a parity tree, error coverage is dependent solely on the amount of cycle slack and not the parity tree size and delay. If the detection delay (parity tree and transition detector) does not equal the desired cycle time slack, the clock for the detector dynamic ORs (now Clk$'$) can be shifted relative to the main clock by δ. This could be useful if, for instance, the designer wishes to decrease the cycle time (sacrificing error coverage) while maintaining parity tree size for maximum area efficiency. In either case, for maximum error coverage, Clk$'$ is shifted to rise immediately after the output of the transition detector has settled. Equation (1) can be used to calculate this optimal skew, where a positive δ indicates a delayed Clk$'$. $MaxDel$ is the maximum delay of the circuit only, $DetDel$ is the detection delay, and T is the cycle time.

$$\delta = (MaxDel + DetDel) - T \tag{1}$$

$$(DetDel + OrDel) \leq \frac{T}{2} + \delta \tag{2}$$

Note that for correct fault detection, the error signal output from the dynamic logic must become high before Clk$'$ falls and the dynamic logic is precharged. This constraint is shown in Equation (2). If not met, transitions occurring at the very end of the main clock cycle may go undetected. As shown, increasing δ allows a larger detection delay. Also, consider the case in which certain parity trees are smaller and have a shorter delay than the maximum size. The error coverage for flip-flops connected to such trees may be decreased unless padding is added to tree outputs to maintain the time delay. Thus, similarly sized trees are ideal. For trees where the number of inputs is not a power of two, buffers can also be inserted balance delays.

C. Flip-Flop Selection

The manner in which flip-flops are chosen for parity groups is highly dependent on the designer's intentions and the characteristics of the circuit to be protected. It is possible to haphazardly group flip-flops, but doing so may not result in an optimal error coverage vs. overhead tradeoff. In addition, some circuits may have very similar delay paths, making it theoretically possible for simultaneous even transitions to occur in a parity tree and be masked. To avoid this situation, extra care can be taken when choosing flip-flop groups.

In the simplest case, we use the following methodology (adapted from [6]) to rank flip-flops in terms of preference for protection:

1) An initial simulation experiment is performed, with a number of faults injected into the combinational logic of the target circuit.
2) A list is kept to track which flip-flops were corrupted by each fault causing an error at a primary output.
3) Once the desired number of faults have been simulated, a total is calculated indicating the number of times each flip-flop was corrupted.
4) The flip-flop with the highest total is chosen, and all faults corrupting this flip-flop are removed from the list.
5) Totals are recalculated, and this process is repeated for each remaining flip-flop.

Note that this is the same procedure used to rank flip-flops for protection using shadow latches (our comparison baseline). For parity, the ranked list of flip-flops is divided into groups depending on the intended parity tree size. Groups are not formed across pipeline boundaries in pipelined circuits.

If concurrent even transitions impose a significant penalty in terms of error coverage, we propose two possible solutions. First, parity group size can be decreased, incurring a slight increase in power and area overhead due to the cost of additional transition detectors and OR logic. This decreases the likelihood of multiple transients propagating through the same tree. A second option is to purposely create disjoint parity groups based on the fault injection data used to create flip-flop rankings. By disjoint, we mean that no two flip-flops in the group ever latched transients from the same injected fault in the initial simulation. Disjoint groups are formed using the following method:

1) From all ungrouped flip-flops, find those that have no conflicts with the group currently being formed. This is done by excluding all flip-flops that were ever corrupted simultaneously with flip-flops in the group being formed.
2) Of the flip-flops with no conflicts, if any, select the one with the highest fault total.
3) If a flip-flop is found, all faults corrupting it are marked as detected in the fault list (removed from totals) and totals are recalculated.
4) If no matches were found, or the maximum group size is reached, a new group is started.

D. SEU Protection

We acknowledge that protecting against upsets in flip-flops may be as important or more important than protecting combinational logic. Our scheme, however, may be used to supplement many methods for flip-flop protection. For instance, a design relying on hardened latches as presented in [2] could be augmented with our technique for logic protection. It should also be noted that our design does provide a level of SEU detection for the master latch, which can be maximized by increasing the duty cycle of the detector clock so that detection is disabled just before data reaches the transition detector from the newly-transparent master latch.

If a SEU occurs in a slave latch near the end of the cycle, such that the edge from the upset arrives during the slack period monitored by our detection circuitry in the next stage, it will be detected exactly as if it were a transient. Also, if the circuit being protected has a minimum delay greater than or equal to half of the cycle time minus the slack period, then a SEU in a slave latch will never be able to reach the circuit output before the detectors are enabled. Of course, if the minimum delay is greater than half the cycle time, slave latches may not be required at all.

IV. Experimental Results

Simulations were performed using an event-driven gate-level simulator with timing parameters for 65nm technology. Gate-level simulation is much faster than circuit-level simulation, and allows timely analysis of larger networks using statistical fault-injection. This method inserts transients into a circuit randomly in time and space. For each injected fault, the circuit input vector is randomly generated. A time offset is chosen at random within the cycle, and a gate to strike is also randomly selected using an area model based on the drain areas of all gates. Transient duration is determined

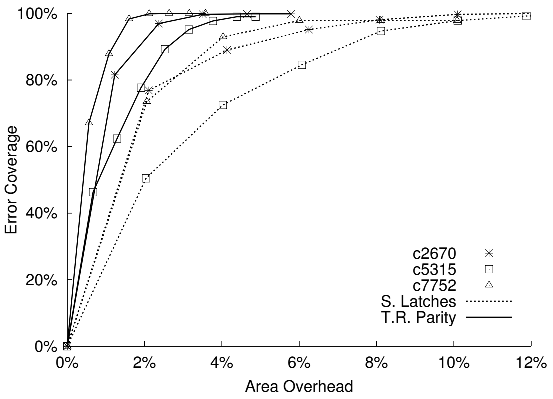


Fig. 3. Error coverage with respect to area overhead for some ISCAS '85 benchmarks.

using a probability distribution function generated from SPICE simulations, as described in [6]. Depending on these parameters, an injected fault may 1) be logically masked, 2) be timing-window masked, or 3) corrupt an output. Note that our simulator does not model electrical masking. Since electrical masking is present in protected and unprotected circuits, we believe the overall SER reduction to be accurate.

Area comparisons of different SET protection techniques were based on transistor count. Detector dynamic OR gates had a maximum of 8 inputs, with a larger number of inputs requiring a tree. We also performed power simulations of protected and unprotected circuits. Dynamic power was estimated using the procedure in [20], in which the switching capacitance for each node is estimated. This allows the calculation of energy for each transition. Dynamic power was averaged over 1,000 random input vectors. Static power calculation was based on the assumption that 30% of total power consumption was due to leakage [21]. Because our power simulations were performed for normal operating conditions (no transients), a portion of the detection circuitry used no dynamic power. To account for this, we computed the static power to device ratio for the unprotected circuit and used this to calculate the static power of the additional devices in the protected version.

TABLE I
CIRCUITS SIMULATED

Name	PIs	POs	Transistors	Max. Delay (ps)	Unprotected Derating
c2670	233	140	8864	1293	0.0254
c5315	178	123	16298	1372	0.0220
c6288	32	32	10752	2614	0.0590
c7752	207	108	19456	1094	0.0271
z80	34	563	49420	755	0.0448
fp-add	162	102	52804	2123	0.0204
fp-mul	164	83	135742	2372	0.0321

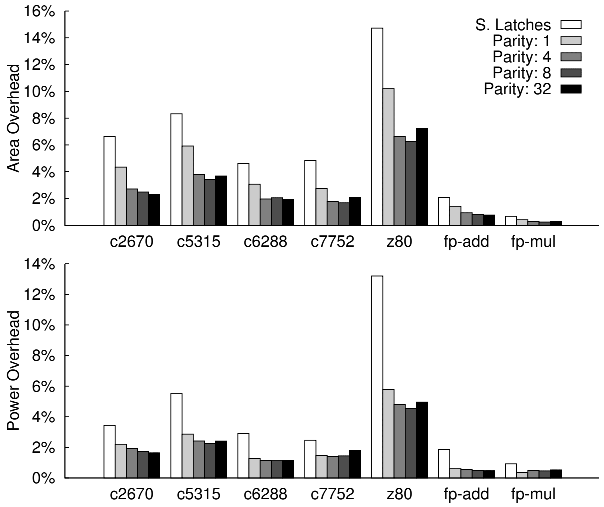


Fig. 4. Area and power overheads with shadow latches and time redundant parity, derating ≤ 0.001.

Table I shows the circuits simulated. All circuits beginning with "c" are combinational ISCAS '85 benchmarks. The circuit labeled "z80" is a 3-stage parallel instruction decoder for the z80 architecture, as studied in [22]. The circuits "fp-add" and "fp-mul" are an OpenSPARC [23] combinational floating-point adder and multiplier. All circuits were augmented with flip-flops at their primary outputs. The unprotected derating shown refers to the fraction of faults injected into the circuit's unprotected combinational logic that caused an error at a primary output. For instance, the derating for c2670 is 0.0254, implying that of 60,000 injected faults (the total for each experiment), 1,524 resulted in errors. The remaining fraction of faults were either logically or timing-window masked. In calculating the derating for our simulations of protected circuits, the only faults considered errors are those that corrupt a primary output and remain undetected.

Simulation included two phases. In the first phase, statistical fault injection (SFI) was performed on the unprotected circuit. Based on this simulation, flip-flop rankings and parity groups were created. In the second phase, the SFI experiment was repeated with a different random seed to evaluate the protected circuit. Two cases were simulated. In the first, shadow latches were added to protect flip-flops through double sampling, as shown in Figure 1. In the second case, flip-flops were connected to parity trees in the manner described in this paper. For each injected fault causing an error, the best-ranked detector able to detect the fault was noted. This methodology allowed us to find an accurate relationship between error coverage and area overhead. Figure 3 shows error coverage with respect to area overhead using 1) shadow latches and 2) time-redundant parity for some ISCAS '85 benchmark circuits. Sixteen-input parity trees were used for time redundant parity, and similar results were produced for all circuits.

To compare the overheads of time redundant parity and shadow latches, we chose a target derating for the circuit and added just enough detectors (in ranked order) to meet the target. Figure 4 shows the area and power overheads of protecting each circuit to reduce its derating to 0.001. This constitutes an average error coverage of 96% and a maximum SER reduction of 59X in the case of the c6288 benchmark. For time redundant parity, the legend indicates the number of parity tree inputs. "Parity: 1" does not use parity trees and instead augments individual flip-flops with transition detectors. All protected circuits use the same cycle slack.

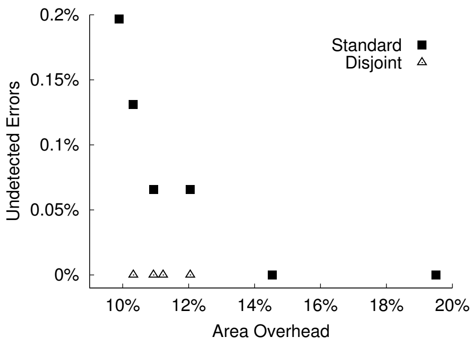


Fig. 5. Example of using disjoint groupings to protect the c2670 benchmark.

As expected, continued increases in parity tree size result in diminishing power and area savings. This can be partly attributed to the very low cost of the dynamic OR gates used. If this cost were increased, perhaps by further limiting the number of gate inputs, larger parity trees would be more advantageous. Also, note that in some cases, larger parity trees require increased overhead to reach the desired derating. This is because larger trees increase the possibility of simultaneous even transients in a parity group, requiring additional trees to make up for these undetected faults. Of course, the effect is emphasized because our simulations consider each parity tree as a whole detector unit. This course-grained approach means that, especially for larger trees, the indicated area overhead may actually provide a higher level of protection than required.

Though we found reducing parity tree size to be the most effective technique for eliminating masked faults to reach an intermediate derating, this approach is not necessarily the best when a very high level of reliability is desired. In this case, when the majority of flip-flops are connected to detectors, our algorithm for creating disjoint parity groups can be applied. Figure 5 shows different configurations protecting all flip-flops from transients in the c2670 benchmark using time redundant parity. The points marked as "Standard" demonstrate the expected tradeoff between the undetected percentage of error-causing faults and area overhead. The configuration with the most undetected errors and lowest area overhead uses 32-input parity trees, while using only transition detectors is at the opposite extreme. Points for intermediate group sizes are also shown. In addition, the plot includes results for disjoint groupings with various maximum tree sizes, showing the effectiveness of this algorithm for eliminating masked transitions in parity trees while maintaining the area savings from larger parity groups. Of course, the effectiveness of the algorithm is highly dependent on the output corruption patterns of the circuit being protected. It is also possible that imposing parity grouping constraints could require additional area overhead due to wiring complexity, but this would be difficult to estimate at the gate level.

V. Conclusion

The logic SER is a growing problem that requires a cost-effective solution. In this paper, a novel technique for single-event transient detection is presented to address this problem. Our technique relies on inexpensive parity trees to compact combinational output SETs, and requires significantly less power and area overhead than previously proposed time redundant techniques that rely on shadow latches. Our results show average savings in power and area overheads of more than 55% without sacrificing error coverage.

Acknowledgment

This work was supported in part by National Science Foundation grants (CCF-0702272, CCF-0953603, and CCF-1016262) and generous gift grants from Microsoft and AMD.

References

[1] S. Mukherjee, J. Emer, and S. Reinhardt, "The soft error problem: an architectural perspective," in *Proc. Int. Symp. High-Performance Computer Architecture*, 2005, pp. 243–247.

[2] M. Nicolaidis, R. Perez, and D. Alexandrescu, "Low-cost highly-robust hardened cells using blocking feedback transistors," in *Proc. IEEE VLSI Test Symposium*, 2008, pp. 371–376.

[3] M. Imhof, H.-J. Wunderlich, and C. Zoellin, "Integrating scan design and soft error correction in low-power applications," in *Proc. Int. On-Line Testing Symp.*, 2008, pp. 59–64.

[4] S. Mitra *et al.*, "Robust system design with built-in soft-error resilience," *Computer*, vol. 38, no. 2, pp. 43–52, Feb. 2005.

[5] P. Shivakumar *et al.*, "Modeling the effect of technology trends on the soft error rate of combinational logic," in *Proc. Dependable Systems and Networks*, 2002, pp. 389–398.

[6] E. L. Hill, M. H. Lipasti, and K. K. Saluja, "An accurate flip-flop selection technique for reducing logic SER," in *Dependable Systems and Networks*, 2008, pp. 128–136.

[7] S. Mitra and E. McCluskey, "Which concurrent error detection scheme to choose ?" in *Proc. IEEE Int. Test Conf.*, 2000, pp. 985–994.

[8] A. Dutta and N. A. Touba, "Synthesis of nonintrusive concurrent error detection using an even error detecting function," in *Proc. Int. Test Conf.*, 2005, pp. 1059–1066.

[9] A. Nieuwland, S. Jasarevic, and G. Jerin, "Combinational logic soft error analysis and protection," in *Int. On-Line Testing Symp.*, 2006.

[10] R. Rao, D. Blaauw, and D. Sylvester, "Soft error reduction in combinational logic using gate resizing and flipflop selection," in *Proc. Int. Conf. Computer-Aided Design*, 2006, pp. 502–509.

[11] K. Mohanram and N. A. Touba, "Cost-effective approach for reducing soft error failure rate in logic circuits," in *ITC 2003*, pp. 893–901.

[12] S. A. Seshia, W. Li, and S. Mitra, "Verification-guided soft error resilience," in *Design Automation and Test in Europe*, 2007, pp. 1–6.

[13] M. Nicolaidis, "Time redundancy based soft-error tolerance to rescue nanometer technologies," in *Proc. VLSI Test Symp.*, 1999, pp. 86–94.

[14] P. Elakkumanan, K. Prasad, and R. Sridhar, "Time redundancy based scan flip-flop reuse to reduce SER of combinational logic," in *Proc. Int. Symp. Quality Electronic Design*, 2006, pp. 619–624.

[15] D. Ernst *et al.*, "Razor: a low-power pipeline based on circuit-level timing speculation," in *MICRO 2003*, pp. 7–18.

[16] S. Mitra *et al.*, "Combinational logic soft error correction," in *Proc. IEEE Int. Test Conf.*, 2006, pp. 1–9.

[17] L. Anghel and M. Nicolaidis, "Cost reduction and evaluation of a temporary faults detecting technique," in *Design Automation and Test in Europe*, 2000, pp. 591–598.

[18] K. Bowman *et al.*, "Energy-efficient and metastability-immune resilient circuits for dynamic variation tolerance," *IEEE J. Solid-State Circuits*, vol. 44, no. 1, pp. 49–63, Jan. 2009.

[19] D. Rossi, M. Omaña, and C. Metra, "Transient fault and soft error on-die monitoring scheme," in *DFT 2010*, pp. 391–398.

[20] N. S. Kim *et al.*, "Microarchitectural power modeling techniques for deep sub-micron microprocessors," in *Proc. Int. Symp. Low Power Electronics and Design*, 2004, pp. 212–217.

[21] K. Aygün *et al.*, "Power delivery for high-performance microprocessors," *Intel Technology Journal*, vol. 9, no. 4, pp. 273–283, Nov. 2005.

[22] E. L. Hill and M. H. Lipasti, "Logic soft errors in a parallel CISC decoder," in *Workshop on Silicon Errors in Logic-System Effects*, 2010.

[23] I. Parulkar *et al.*, "OpenSPARC: An open platform for hardware reliability experimentation," in *Workshop on Silicon Errors in Logic-System Effects*, 2008.