2004－SLDM－117（6）
2004／12／1

A Cost-effective Technique to Mitigate Soft Errors in Logic Circuits

Luong D. HUNG[†], Masanori TAKADA[†], Yi GE[†], and Shuichi SAKAI[†]

[†] Graduate School of Information Science and Technology, The University of Tokyo

Abstract The soft error rates (SER) in logic circuits increase quickly as devices scale. Existing techniques to mitigate soft errors in logic circuits often incur large overheads. In this work, we propose a 'lightweight' technique that detects soft errors in logic circuits, utilizing the concept of temporal sampling. The technique adds some modifications to the conventional pipeline to allow data to be sampled twice in time and compared for integrity. The area, power, and timing overheads of modifying a 32-bit multiplier to support the technique are respectively 19.3%, 7.6%, and 6.4%. Comparing to existing soft error detection circuit techniques, our technique incurs lower overheads. The technique is also applicable in scaled process technologies.

Key words Soft error, SER, logic circuit, pipeline, flipflop.

1. Introduction

Energetic particles, such as neutron particles from cosmic rays or alpha particles in packaging materials, when pass through a semiconductor device could collide with the silicon nuclei and generate multiple electron-hole pairs [1] [2]. Transistor nodes can collect these charges and sufficient amount of accumulated charges can invert the state of a gate, a latch, or a SRAM cell and cause errors. Such an error is called *soft error* or *transient error*. Shrinking of device geometries, lowering of supply voltages make the devices more susceptible to soft errors. Moreover, the number of gates included in a chip increases rapidly, magnifying the system Soft Error Rate (SER) [3].

Traditionally, memories have been the main victims of soft errors. While memories can be effectively protected from soft errors using rather simple techniques like error detecting/correcting codes, interleaving layout of memory words, mitigating soft errors in logic circuits is more challenging. Common circuit techniques to mitigate soft errors in logics usually impose high costs of implementation [4] [5]. The high costs are intolerable in many commonplace applications and soft errors in logics are simply overlooked in such cases.

However, SER in logic circuits increases rapidly with scaling trend. Shivalkurmal et al. [6] found out that the SER of individual logic circuits increases exponentially as devices shrink. Similar effect is also claimed by other work [3]. Ignoring soft errors in logic circuits becomes increasingly not a viable option.

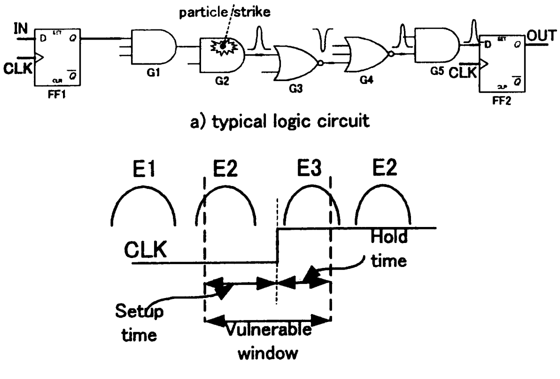
Our work aims to provide a cost-effective technique to detect soft errors in logic circuits. Our technique involves replication of memory elements and clock to allow the results of logic circuits to be sampled twice. The second sampling is delayed after the first sampling by a time interval sufficiently larger than the erroneous pulse width of a soft error. The results of the two samplings are then compared and any mismatch in the results indicates that soft error has happened and disrupted the results. We verified that the technique can be implemented with moderate overheads.

The rest of the paper is organized as follows. Section 2. explains how soft errors occur and affect the results of conventional logic circuits. Section 3. describes in detail how a pipeline can be modified to support soft error detection. Section 4. presents evaluation results. Section 5. discusses related work. Finally, Section 6. concludes the paper.

2. Soft Error in a Conventional Pipeline

Figure 1-a illustrates a typical logic circuit, which represents a single pipeline stage. The logic circuit is sandwiched between two memory elements. Static, edge-triggered circuit style is assumed here and the *vulnerable window* of a flipflop is defined as the time interval locates at the sampling edge of the clock, as shown in Figure 1-b. The durations from the clock sampling edge to the left side and the right side of the vulnerable window are respectively equal to the setup time and hold time of the flipflop. The input from the first flipflop (FF1) is sampled and then passed to the logic on a clock edge. Proper design should make sure that the result computed by the logic is available at the beginning of the vulnerable window of the second flipflop (FF2) and stays stable during the window for being correctly sampled.

Let us consider the situation when a particle strikes on the AND gate (G2) and temporarily upsets its output. An erroneous pulse appears at the gate output. The pulse could propagate and modify the outcomes the later gates. If the erroneous pulse finally reaches at the input of FF2 during its vulnerable window (E2 and E3 in Figure 1-b), FF2 may sample the incorrect data.



a) typical logic circuit

b) vulnerable window

Fig. 1 Typical sequential logic circuit.

3. Proposed Technique

3.1 Concept

Our technique relies on the property that a static logic circuit not containing any memory element, when struck by a particle, could generate erroneous output but the effect is *transient*. The logic is able to recover the right value by itself after some amount of time. It can be explained as follows. The induced charges at diffusion node (source or drain node) erroneously change the output of the gate. However, as long as the input voltage is unchanged, the diffusion node then collects the charges from voltage source (GND or VDD) and gradually swings back to the original voltage. The amount of time for recovering is roughly equal to the width of the spurious pulse generated due to the particle strike. The typical pulse widths, as we will show later, are in the order of few hundred pico-seconds at current process technologies and becomes smaller as devices scale.

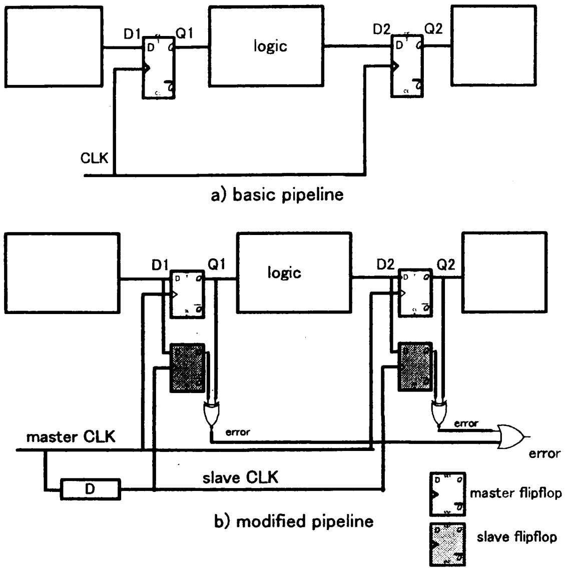
Noteworthy, the property described above is true for static logics but does not applies to dynamic logics. In dynamic logics, if the output of a gate erroneously discharges from high to low due to soft error during the evaluation phase, the gate can not recover back to its high voltage until precharge phase. Although dynamic logics are intrinsically fast, they consume more power, are more susceptible to noise, and difficult to design than static logics [7]. Except the cases where performance is critical, designers often favor static logics than dynamic logics. We focus on soft error detection in static logics in this paper.

The governing idea of our technique is to sample the output of a logic circuit twice. The first sampling is followed by the second sampling after some amount of delay. By setting the delay between the two samplings sufficiently larger than the erroneous pulse width, if any soft error happens and disrupts the results of the first sampling, the result will recover from the error by the time the second sampling takes place. The results of the two samplings are then compared. Any mismatch in the results indicates that a soft error has been occurred. Next, we explain how a pipeline can be modified to support this mechanism.

3.2 Pipeline Supporting Soft Error Detection

Figure 2-a shows a conventional pipeline. For simplicity, the logic here contains one input and one output though we could easily extend to the cases of multiple inputs and outputs. Figure 2-b shows the modified pipeline that supports soft error detection. The modified pipeline has two clock signals, namely ***master clock*** and ***slave clock***, and two types of flipflops, namely ***master flipflops*** and ***slave flipflops***. Master clock and master flipflops already exist in conventional pipeline while slave clock and slave flipflops are newly added. The slave clock, which has the delay D in related to the master clock, goes in and controls slave flipflops.

The result produced by the logic is sampled twice. The first sampling occurs at the rising edge of the master clock where the result is sampled into the master flipflop and is ***immediately*** available to the following pipeline stage. The second sampling happens at the rising edge of the slave clock and the result is stored in the slave flipflop. By providing the time delay D between the master clock and the slave clock large enough, we could guarantee that if a transient error results in incorrect output in the first sampling and the same transient error dies away at the time the second sampling taking place. Once such condition is satisfied, we can detect the possible error by comparing the two results. If the two results are identical, no error has happened. Mismatch in the results indicates that error has happened and corrupted data. Comparison of the results is carried by the XNOR, OR gates as indicated in Figure 2-b.



a) basic pipeline

b) modified pipeline

Fig. 2 Conventional & Modified Pipeline.

Next, we are going to describe the constraints the modified pipeline has to satisfy. We come up with two constraints: the first one constrains the time delay D of the slave clock related to master clock, and second one constrains the propagation time of the shortest path in the logic.

Constraint for D

Proper setting of D, which is the amount of delay of the slave clock in related to the master clock, is very important. We derive the following formula that determines the lower bound of D.

$$D \geq t_{hold} + W + t_{set} + t_{skew} \qquad (1)$$

Here W is the maximum width of erroneous pulses caused by particle strikes. t_{hold}, t_{set} are respectively the holding time, setup time and clock-to-Q delay of flipflops. t_{skew} is the clock skew.

Equation 1 can be understood by consulting Figure 3. The latest erroneous pulse that may disrupt the output of the first sampling begins after the rising edge of the master clock by an amount of time equal to flipflop hold time. For correct detection, the error must recover before the rising edge of the slave clock by an amount of time equal to the setup time of the flipflop. The minimum value of D is therefore the sum of the erroneous pulse width, the clock skew, the hold time and the setup time of the flipflops.

Min-timing Constraint

Min-timing constraint guarantees that a flipflop already finishes the second sampling of the old data by the time the new data arrives at its input. Figure 4-a illustrates the case where the min-timing constraint is violated. In this example, Q1 is ready by the time t_{cq} after the rising edge of the master clock (t_{cq}, or clock-to-Q delay, of a flipflop is the delay from the rising edge of the clock until the sampled data is available at the flipflop output). Q1 is then fed into the logic and the resulting data is available at D2 as soon as T_{min} later, where T_{min} is the minimum propagation time. The new data arrives at D2 before the rising edge of the slave clock and overwrites the old input of FF2. So in the second sampling, FF2 samples the new input instead of the old input, which is intended for.

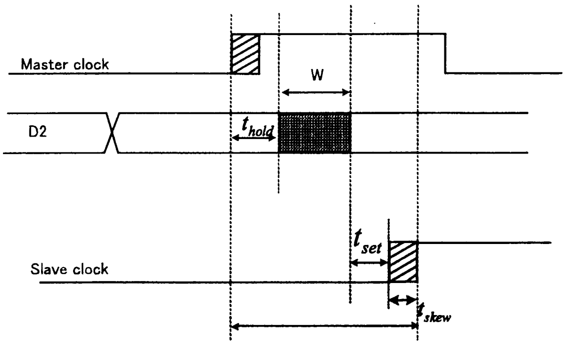
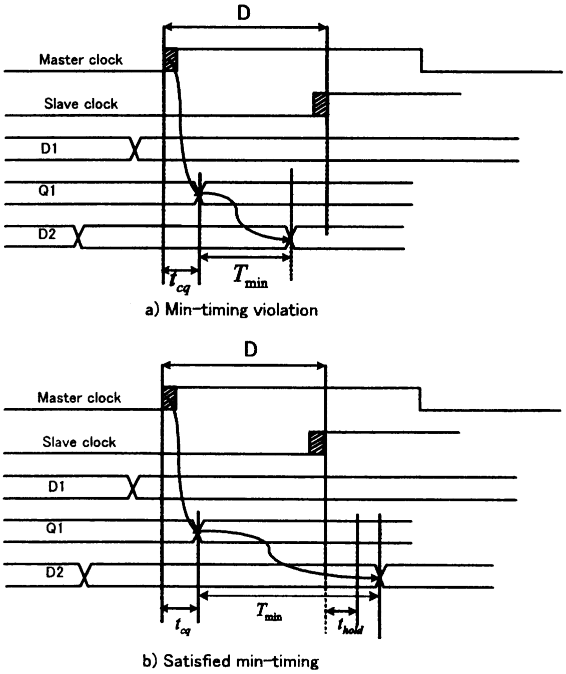


Fig. 3 Determine the delay D of slave clock related to master clock.



a) Min-timing violation

b) Satisfied min-timing

Fig. 4 Min-timing constraint in modified pipeline.

On the other hand, Figure 4-b illustrates the case where min-timing constraint is satisfied. In this case, when the

FF2 has already finished the second sampling by the time the fastest signal arrives at its input.

The min-timing constraint can be written as follows.

$$t_{cq} + T_{min} \geqq D + t_{hold} \quad (2)$$

and rearranged as

$$T_{min} \geqq D + t_{hold} - t_{cq} \quad (3)$$

Buffers could be inserted into paths that are too short to create enough propagating time required by Equation 3.

3.3 Consideration

3.3.1 Overheads

Area overhead. A modified pipeline requires the same number of slave flipflops as master flipflops, therefore, doubles the number of flipflops in conventional pipeline. Additional clock, the slave clock, is also required. We could choose to distribute slave clock globally, or to generate it locally from the master clock by taking the master clock and inserting some buffers to create right amount of time delay, as dictated by Equation 1. Another overhead is due to additional gates used for comparing the results. These overheads depend on the numbers of inputs or outputs but do not depend on the complexity of the logic. Finally, there may be overhead due to buffers inserted to satisfy the min-timing constraint (Equation 3).

Timing overhead. Though the modified pipeline imposes new min-timing constraint (Equation 3), it does not impose any constraint relating to max-timing. Clock cycle is therefore almost unaffected by modifying the pipeline. In the modified pipeline, computation and synchronization of data between pipeline stages are coordinated by the master clock. Master flipflops can pass the data of the first sampling to the following pipeline without waiting until the data is sampled again and checked. Second sampling and checking of data produced by one stage occurs in parallel with computation of the next stage. The time overhead can be largely hidden in multi-stage pipeline. The checking overhead appears explicitly only at the final stage where we must wait until checking completes before committing the results.

Power overhead. Because power consumption becomes increasingly important in VLSI designs, it should be considered as one important measure when evaluating the costs. The logic between the pipeline flipflops and its switching activity is almost unchanged by modifying the pipeline. The power consumption may increase due to samplings of slave flipflops and switching of the slave clock. Whether the power overhead is small or large depends on the ratio of the amount power consumed in the logic and the power consumed by flipflops. If the logic is fairly large and consumes most of the power, the power overhead of our technique can be small.

3.3.2 Error Detection Coverage

The modified pipeline described above can detect the soft errors caused by a single particle strike, regardless of whether a single or multiple adjacent nodes inside the logic are upset by the same strike. If multiple particle strikes occur in the logic within a same clock cycle, the logic may require more time to recover from the errors. The master and slave flipflops could sample the same incorrect data, leaving the soft errors undetected. However, particle strikes are infrequent events and the chance of multiple particle strikes occurring inside a clock cycle is much smaller than that of single particle strike [2]. So not covering multiple strikes only slightly impacts the error detection coverage.

Although we mainly focus on soft errors that occurs in the logic, a particle could strike on a flipflop and modify its content. Replication of the flipflops in our technique also enables such kind of errors to be detected.

4. Evaluation

The effectiveness of the technique depends largely on the widths of soft error pulses (W). We first carried circuit simulation to measure the values of W. Sharp currents that model the currents induced by particle strikes are injected into an inverter chain. The widths of the voltage responses are measured subsequently. The values of parameters characterizing the currents induced by atmospheric neutrons at various process technologies are obtained from [8].

The result is shown in Figure 5. At any given technology, the erroneous pulse width varies for individual strikes. The pulse width scales well with technology scaling. The scaling of erroneous pulse widths is advantageous because it allows the technique to be applicable at small process technologies without introducing excessive time overhead.

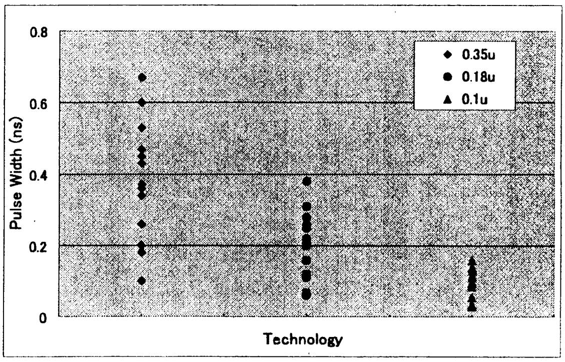


Fig. 5 Erroneous pulse widths at different process technologies.

We next implemented a 32-bit, five-stage, non-Booth recorded Wallace-tree multiplier. We then modified the origin multiplier to enable it to detect soft errors, following the technique described so far. In the modified version of the multiplier, the master clock and the slave clock are both distributed globally. These circuits are implemented using a standard cell library basing on Hitachi 0.18 um process technology. Table 1 shows the values of setup time, hold time, and clock-to-Q delay of flipflops found in the library.

setup time (t_{set})	0.16 $\sim$ 0.17 ns
hold time (t_{hold})	0.06 $\sim$ 0.065 ns
clock-to-Q delay (t_{cq})	0.18 $\sim$ 0.348 ns

Table 1 Setup time, hold time, and clock-to-Q delay of flipflops in the standard cell library.

The maximum erroneous pulse width (W) in 0.18 um technology is set to 0.4 ns basing on the results in Figure 5. The clock skew is set to 50 ps, which is an achievable value in real design [9]. D and T_{min} are respectively set to 0.64 ns and 0.53 ns, which satisfy Equation 1 and Equation 2 even in the worst case.

	Before	After	Overhead
Area	0.264 mm^2	0.315 mm^2	19.3 %
Power	1.18 W	1.27 W	7.6%
Timing	4.5 ns|22.5 ns	4.58 ns|23.94 ns	1.7%| 6.4%

Table 2 Area, power, and timing before and after modificating the 32-bit multiplier.

Table 2 shows the overheads of the circuit before and after modification for supporting soft error detection. The area, derived from the final layouts, increases by 19.3%. The power consumption, calculated by extracting nest lists from the layouts and feeding them with randomly generated inputs and then averaging, increases by 7.6%. The increases in area and power consumption are caused by switching of slave clock, slave flipflops as well as additional buffers inserted to satisfy the min timing constraint.

The timing has two values. The first one is the clock cycle. The second is the total latency when the inputs enter in the first stage until the confirmed outputs are available at the fifth stage ($\sim$ *clock cycle* × *number of stages* + *2nd sampling, checking time at the final stage*). Clock cycle increases slightly in modified circuits. Because the time overheads of the second sampling and checking of the first to the fourth stages can be overlapped, the increase in total latency is small (6.4%).

5. Related Work and Discussion

Soft errors due to particle strike are long-recognized phenomena and have drawn a lot of research efforts. Traditional architectural transient-fault tolerant techniques such as triple modular redundancy (TMR) or double modular redundancy (DMR), or self-checking logics incurs high costs [10]. Several interesting architectures for transient-fault tolerance in the context of superscalar processors [11], simultaneous multithreading processors [12], or chip multiprocessors [13] have been proposed in recent years. While performance degradations are proved to be small, these techniques still require large power overhead and introduce new levels of design complexity.

Common circuit techniques to detect transient errors in logics are parity prediction [5], or residue code checking [4]. According to [5], the overheads for implementing parity prediction in a 32-bit multiplier are 63.7% in area, 19% in latency, and 65% in power. The area overhead for implementing residue code checking in 32-bit multiplier as high as 20.7% is reported in [4]. The evaluation results show that our technique achieves comparable or lower overheads compared to these techniques.

The most related work is done in [14], which pioneers the idea of temporal sampling for eliminating soft errors. In [14], the data is sampled three times then voted for majority. Although such technique allows not only error detecting but also error recovering, it requires three time more flipflops, three clock tracks as well as voting circuits. More importantly, the result is available to next pipeline only after samplings and majority voting finish, which increases the latency considerably. Our technique, on the other hand, samples data only twice and the data is passed to next pipeline as soon as the first sampling finishes so that the overhead of the second sampling can be hidden in a multi-stage pipeline. In [15], the authors make use of temporal sampling to measure the widths of transient pulses.

6. Conclusion

In this paper, a circuit technique for soft error detection in logics is proposed. The technique adds some modifications to conventional pipeline to allow the data to be sampled twice. The second sampling is delayed after the first sampling by a time interval sufficiently larger than the erroneous pulse width of a soft error. The results of the two samplings are compared and any mismatch in the results indicates that a soft error has happened and corrupted the data.

The timing constrains imposed by the technique were derived. Real circuit design was carried to evaluate the overheads of the technique. The area, power, and timing overheads of modifying a 32-bit multiplier to support the technique are respectively 19.3%, 7.6%, and 6.4%. Comparing to existing soft error detection circuit techniques, our technique incurs lower overheads. The technique is also applicable in scaled process technologies.

Acknowledgments This research is partially supported by Grant-in-Aid for Fundamental Scientific Research B(2)

#13480077 #16300013 from Ministry of Education, Culture, Sports, Science and Technology Japan, and by Semiconductor Technology Academic Research Center (STARC) Japan, CREST project of Japan Science and Technology Corporation, and by 21st century COE project of Japan Society for the Promotion of Science, and VLSI Design and Education Center (VDEC), the University of Tokyo in collaboration with Cadence Design Systems, Inc, Hitachi Ltd, Mentor Graphics, Inc, and Synopsys, Inc.

References

[1] R. C. Baumann: "Soft Errors in Advanced Semiconductor Devices-Part I: Three Radiation Sources", IEEE Transactions on Device and Materials Reliability, **1**, 1, pp. 17–22 (2001).

[2] J. F. Ziegler: "Terrestrial commis rays", IBM Journal of Research and Development, **40**, 1 (1996).

[3] R. C. Baumann: "The Impact of Technology Scaling on Soft Error Rate Performance and Limits to the Efficacy of Error Correction", International Electron Devices Meeting, pp. 329–332 (2002).

[4] U. Sparmann and S. M. Reddy: "On the Effectiveness of Residue Code Checking for Parallel Two's Complement Multipliers", IEEE Transactions on Very Large Scale Intergration Systems, **4**, 2, pp. 227–239 (1996).

[5] K. S. Papadomanolakis, et al.: "The Effect of Fault Secureness in Low Power Multiplier Design", International Workshop on Power and Timing Modeling, Optimization and Simulation, pp. 26–28 (2001).

[6] P. Shivakumar, M. Kistler, et al.: "Modeling the Effect of Technology Trends on the Soft Error Rate of Combinational Logic", International Conference on Dependable Systems and Networks (2002).

[7] A. P. Chandrakasan, S. Sheng and R. W. Brodersen: "Low-Power CMOS Digital Design", IEEE Journal on Solid State Circuits, pp. 473–484 (April, 1992).

[8] P. Hazucha and C. Svensson: "Impact of CMOS Technology Scaling on the Atmospheric Neutron Soft Error Rate", IEEE Transactions on Nuclear Science, **47**, 6, pp. 2586–2594 (2000).

[9] D. W. Bailey and B. J. Benshneider: "Clocking Design and Analysis for a 600-MHz Alpha Microprocessor", Journal on Solid State Circuits, pp. 1627–1633 (1998).

[10] P. K. Lala: "Self-checking and Fault-tolerant Digital Design" (2001).

[11] T. M. Austin: "DIVA: a reliable substrate for deep submicron microarchitecture design", International Symposium on Microarchitecture (1999).

[12] S. K. Reinhardt and S. S. Mukherjee: "Transient-fault detection via simultaneous multithreading", International Symposium on Computer Architecture, pp. 87–98 (2002).

[13] M. Gomaa, et al.: "Transient-fault recovery for chip multiprocessors", International Symposium on Computer Architecture (2003).

[14] D. G. Mavis and P. H. Eaton: "Soft Error Rate Mitigation Techniques for Modern Microcircuits", International Reliability Physics Symposium, pp. 216–225 (2002).

[15] M. Nicolaidis and R. Perez: "Measuring the Width of Transient Pulses Induced by Ionising Radiation", International Reliability Physics Symposium (2003).