On the Role of Timing Masking in Reliable Logic Circuit Design

Smita Krishnaswamy [†], Igor L. Markov [† ♯], and John P. Hayes[†]
[†] The University of Michigan, EECS Department, Ann Arbor, MI 41809
[♯] Synplicity Inc., 600 W. California Ave., Sunnyvale, CA 94086
{smita, imarkov, jhayes}@eecs.umich.edu

Abstract

Soft errors, once only of concern in memories, are beginning to affect logic as well. Determining the soft error rate (SER) of a combinational circuit involves three main masking mechanisms: logic, timing and electrical. Most previous papers focus on logic and electrical masking. In this paper we develop static and statistical analysis techniques for timing masking that estimate the error-latching window of each gate. Our SER evaluation algorithms incorporating timing masking are orders of magnitude faster than comparable evaluators and can be used in synthesis and layout. We show that 62% of gates identified as error-critical using timing masking would not be identifiable by considering only logic masking. Furthermore, hardening the top 10% of error-critical gates leads to a 43% reduction in the SER. We also propose a more subtle solution, gate-relocation for technologies where wire delay dominates gate delay. We decrease the error-latching window of each gate by relocating it in such a way that path lengths to primary outputs are equalized. Our results show a 14% improvement in SER with no area overhead.

1 Introduction

Soft errors are becoming an important concern for circuit reliability. Traditionally, researchers have resorted to techniques which require high area overhead like triple modular redundancy (TMR) in order to obtain fault-tolerant circuits. However, recently the focus has shifted to less intrusive techniques which increase circuit robustness without requiring complete fault tolerance. For instance, [12] proposes gate hardening to increase reliability with low overhead. In [1], the authors use rewiring to decrease susceptibility to SER. In [9], signatures are used to identify redundancy that can be exploited to increase logic masking. Most of these papers use logical masking to increase reliability. In this paper we show how to make prior techniques timing-aware.

Soft errors are a usually a result of thermal neutrons or external radiation. Examples include cosmic particle hits and secondary effects that disturb the silicon substrate and create a transient charge. Transient charges can be latched as errors if the masking mechanisms fail to stop their propagation. Researchers have identified three mechanisms by which soft errors are mitigated in combinational circuits: logical, electrical and timing masking [17]. *Logic masking* happens when an error occurs in a non-sensitized portion of the circuit, and depends on the circuit's input vector and state. *Electrical masking* occurs when a particle strike does not carry enough energy to propagate. Timing masking occurs when an error reaches a flip-flop at a non-latching portion of the clock.

In this paper we propose a linear-time algorithm in the spirit of static timing analysis (STA) for computing the *error latching window (ELW)* of each gate in a circuit. We also show how to incorporate input-vector dependency into our estimates using logic simulation. Similar to the work in [9], our algorithms can be added to logic and physical design flows to increase design robustness. We illustrate our techniques when selecting critical gates for hardening.

We further observe that physical synthesis techniques like gate relocation, buffer insertion, and gate sizing can be utilized to increase timing masking. These techniques are normally used after placement to obtain timing closure [3]. However, with careful guidance, they can improve timing and reliability simultaneously. We demonstrate here that, as wire delay starts to dominates gate delay, post-placement local gate relocation can result in smaller latching windows with no increase in circuit area.

One advantage of our proposed methods is that they can improve design margins with low overhead in a way that increases performance. If a circuit experiences fewer errors using our techniques, it may be possible to lower the threshold voltage or increase the clock frequency while still meeting reliability goals. In this way, reliability improvements can directly translate into performance improvements. Architectural solutions such as error-detection and rollback schemes are often coarse-grained and incur huge performance penalties by flushing pipelines or resetting states. If these are avoided by guided-synthesis, then the average performance can be improved. The main contributions of this paper are:

- Two efficient algorithms for the static and statistical modeling of timing masking within an SER framework that is meant to guide design flows.
- The application of the SER algorithms to gate hardening for increased reliability.
- SER improvement by means of a novel gate relocation technique that increases timing masking.

The remainder of paper is organized as follows. Section 2 discusses previous work on reliability evaluation and reliability-driven synthesis. Section 4 presents our latching-window computation algorithms and their incorporation into a larger SER evaluation framework. Section 5 describes two strategies for utilizing latching-window masking in synthesis. Section 6 presents empirical results, while Section 7 concludes the paper.

2 Previous Work

In the past two years several reliability evaluators such as FASER [21], MARS-C [11], SERA [22], AnSER [9] and the tools from [16], [5] have appeared in the CAD literature. MARS-C and FASER use symbolic techniques to compute logic and electrical masking with a high level of accuracy. However, FASER and MARS-C scale the final error probability by a factor that depends on the length of the clock cycle. This can be inaccurate because, as we show, the latching window is not a constant fraction of the cycle time and strongly depends on the location of the gates and the logical characteristics of the circuit. AnSER takes a more heuristic approach and uses logic simulation to estimate the SER. It only considers logic masking in order to direct synthesis flow and ignores timing masking. SERA [22] does not compute logic masking, and operates on a per-vector basis (as specified by the user), however it accounts for electrical and timing masking through HSPICE-based fault simulation.

Fault tolerance has been a subject of study since the dawn of digital design. In the 1950's von Neumann derived some theoretical bounds for component error that circuits can reliably sustain. His use of cascading TMR and NAND-multiplexing was insightful in fault-tolerant (FT) design, though impractical due to its large overhead. At the gate level, Tyron [18] proposed quadded logic which involves replacing each gate with four gates such that any single error can be tolerated.

Another class of FT architectures uses time redundancy rather than space redundancy for fault tolerance. In the 1990's Nicolaidis proposed a scheme where three latches sample a signal with small delays between them. A voter is used to decide the correct value of the signal. Since stray glitches tend to have short duration, an erroneous value induced by the glitch will be sampled by a single latch with high likelihood. In the related Razor [7] approach, signals are sampled twice. If an error is found, then a pipeline-based recovery mechanism restores signal correctness.

More recently, some researchers have explored guiding synthesis operations by reliability. Almukhaizim et al. [1] use rewiring to increase reliability. The authors of [9] guide a local transformation technique known as rewriting to replace 4-input cuts in a circuit with more reliable sub-circuits. These authors also identify partially redundant signals, insert a gate computing either the AND or OR of such signals and reroute the fanout cones to go through the output of the newly inserted gate to increase logic masking. Our work builds on techniques in [9] and makes them timing-aware.

3 Logic Simulation and SER

In this section, we review logic simulation-based signatures which were used to model logic masking in [9]. Later, we extend this method to incorporate timing masking as well.

In general, logic masking occurs when a signal is not observed at an output due to the presence of controlling values on side paths. In [9] a test-vector counting method is used to compute the effect of logic masking. The idea is that a soft error is logically propagated when appropriate test vectors are applied to the circuit's inputs. Therefore, the fraction of test vectors for an error at a gate output is an estimate of the probability of error propagation. The test vector count is approximated here by means of a testability measure computed via bit-parallel logic simulation. Testability is assessed as a function of signal probability and observability.

Signal probability is computed in [9] by simulating random input vectors through a circuit's gates in topological order. The collection of output responses at the gates with respect to a set of input vectors is known as a *signature*. More formally, for input vectors $\{v_1, v_2, v_3, \ldots v_k\}$, the signature at a node f is the sequence resultant values at f written $sig(f) = f(v_1)f(v_2)f(v_3)\ldots f(v_k)$.

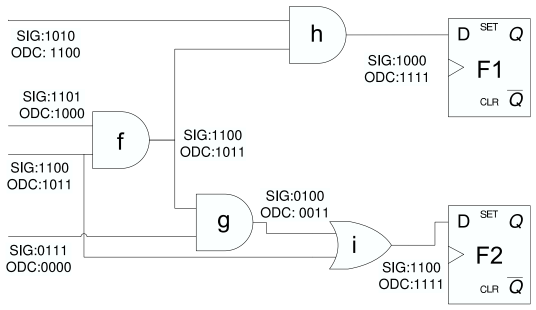

Figure 1: Circuit with signatures and ODCs computed for each signal.

Besides signatures, observability don't-care (ODC) masks are used to estimate signal observability. These correspond to input vectors for which the value of a signal is propagated to a primary output. Observability is computed in reverse topological order by flipping bits in the signature and checking if the resulting signal value is propagated locally. If so, we assess further propagation by examining precomputed ODC masks for signals at fanout points. Corresponding to a k-bit signature $sig(f)$, we define $ODC(f)$ as the k-bit sequence whose ith bit is 0 if input vector v_i is in the don't-care set of f; otherwise the ith bit is 1. Formally, $ODC(f) = v_1 \in care(f), v_2 \in care(f), \ldots, v_k \in care(f)$.

Example 1 *Figure 1 shows a circuit with 4-bit signatures and ODCs. The signatures at the primary outputs are randomly generated and propagated using bit-parallel operations. For instance* $Sig(f) = (1100\ \&\ 1101) = 1100$. *The ODCs are computed in reverse topological order. The ODC of a primary output is* 1111 *since it is always observable. The ODC of f is computed as follows: first the ODC through g is computed, and then the ODC through h is computed and these are ORed together to form the ODC of f. The OR operation accounts for the fact that f is observable if it is observed through g or h. The local ODC of f-via-g is computed by flipping each bit of f and seeing if the effect propagates to g. In the figure this local ODC is* 0111. *This is then ANDed with the pre-computed ODC of g to yield* $ODC_g(f) = 0111\ \&\ 0011 = 0011$. *Similarly the ODC of f through h is computed as* $ODC_h(f) = 1010$. *Finally the ODC of f is given by* $ODC(f) = ODC_h(f) + ODC_g(f) = 1010 + 0011 = 1011$.

Signatures and ODC masks are both stored as bit vectors and computed using bitwise operations. Together the signature and ODC mask provide information about testability. For instance, all bit positions where the signature and ODC mask are 1 correspond to test vectors for a 0-1 fault or glitch at the node. Formally, for a node f the testability of a transient 1-to-0 bit flip is:

$$test_1(f) = \text{ones}\big(sig(f)\&ODC(f)\big)/K$$

Similarly for a 0-to-1 error:

$$test_0(f) = \text{ones}\big(\sim sig(f)\&ODC(f)\big)/K$$

The SER rate, which incorporates logic masking is simply the sum of testabilities at each node weighted by the gate error probabilities $gerr0(f)$ and $gerr1(f)$ for 1-to-0 and 0-to-1 errors, respectively.

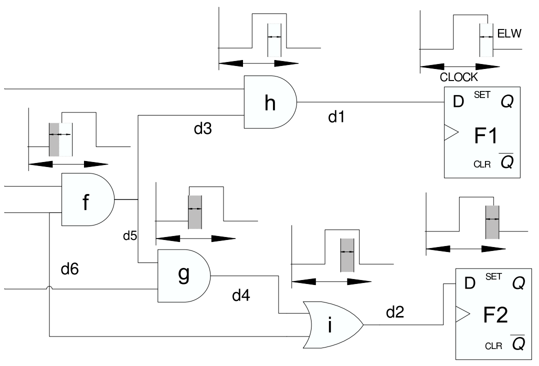

Figure 2: Illustration of error-latching window computation.

4 Error-Latching Windows

This section presents a method for computing error-latching windows (ELWs) for a sequential circuit with edge-triggered flip-flops separated by logic combinational blocks.

4.1 Static Analysis

The timing constraints associated with each edge-triggered D flip-flop are as follows:

- The data (D) input has to receive all data before the setup time T_s preceding the latching clock edge.
- The data input must be held steady for the duration of the hold time, T_h following the latching clock edge.

Soft errors are usually characterized by a transient glitch of duration d that results from a particle strike. If such a glitch is present at the data or clock inputs of a flip-flop during the interval $[T_s, T_h]$, it can result in an incorrect value being latched. If the glitch is present during the setup or hold time, it can prevent a correct value from being latched. Therefore the ELW of the D-flip flop is simply $[T_s, T_h]$.

The ELW for a gate is computed by (1) translating the ELWs of each of its fanout gates backwards by appropriate path delays, and (2) taking the union of the resulting ELWs. In contrast, during static timing analysis we compute only the minimum required time at each gate even though a similar backwards traversal is used. Figure 3 shows the algorithm that computes the union of such intervals. The union of two intervals can result in two separate intervals if the respective intervals are disjoint, or one if the intervals overlap. In general, the latching window for a gate g is defined by a sequence of intervals $ELW(g)[0], ELW(g)[1]\dots$ where $ELW(g)[i]$ refers to the ith interval in the latching window. Each interval $ELW(g)[i]$ is itself described by its start and end times $[S_{gi}, E_{gi}]$.

$$ELW(g) = ([S_{g1}, E_{g1}], [S_{g2}, E_{g2}], \dots [S_{gn}, E_{gn}])$$

Example 2 *Our proposed ELW computation is illustrated by the circuit in Figure 2. Each wire is marked with a delay and each gate i is assumed to have delay $d(i)$. The corresponding ELWs are:*

$ELW(F1) = ELW(F2) = [T_s, T_h]$
$ELW(i) = [T_s - d2, T_h - d2]$
$ELW(g) = [T_s - d2 - d(i) - d4, T_h - d2 - d(i) - d4]$
$ELW(h) = [T_s - d1, T_h - d1]$
$ELW(f) = [T_s - d2 - d(i) - d4 - d(g) - d5, T_h - d1 - d(h) - d3]$
Note that f has a larger ELW than other gates because its two output paths have different delays.

```
compute_ELW(Circuit C)
{  reverse_topological_sort(C);
   for(all latches $l \in C$)
     $ELW(l) = [T_s(l), T_h(l)]$;
   for(all gates $g \in C$)
     for(all fanouts f)
       $ELW'(f) = translate(ELW(f), delay(l,f))$
      $ELW(l) = union(ELW(l), ELW'(f))$;}
```

Figure 3: Algorthm to compute the error-latching windows (ELW) of a circuit.

```
union(ELW(g), ELW(f))
{  for(all intervals ELW(g)[i])
    insert_interval(ELW(g)[i],ELW(f)[1])
  return ELW(f); }
insert_interval(ELW(g)[i],ELW(f)[j]))
{  if($E_{gi} < S_{fj}$)
    return insert_before(ELW(f)[j],ELW(g)[i])
  if($S_{gi} > E_{fj}$)
    if(j == size(ELW(f)))
    return insert_after(ELW(f)[j],ELW(g)[i]);
    else
    return insert_interval(ELW(g)[i],ELW(f)[j+1]);
  $S_{gi} = max(S_{gi}, S_{fj})$;)
  $E_{gi} = min(E_{gi}, E_{fj})$;)
  delete ELW(f)[j];
  return insert_interval(ELW(g)[i],ELW(f)[j]); }
```

Figure 4: Algorithm to compute the union of two ELWs.

We define the *timing masking factor* as the ratio of the ELW to the clock cycle time C. For a node f, the timing masking factor is computed as follows:

$$Tmask(f) = \sum_{i=1}^{n} (E_{fi} - S_{fi})/C$$

Taking timing masking into account, the SER contribution of each gate is computed by scaling the testability and error probability by *Tmask*.

$$SER(C) = \sum_{g \in C} \big(test1(g)gerr0(g) + test0(g)gerr1(g)\big)Tmask(g) \tag{1}$$

4.2 Statistical Interval Weighting

The latching windows computed in the previous section were a result of static analysis. Therefore, some intervals (or portions of intervals) correspond to paths that are not traversed frequently. Our aim is to weigh each interval in the ELW by the probability that an error occurring within the interval gets latched. In order to compute such a probability, we use bit-parallel logic simulation. Recall from Section 3 that the ones count of the signature of a signal is used as a measure of signal probability and the ones count of the ODC mask is used as a measure of signal observability. Together these measures give an estimate of the testability of the associated stuck-at fault.

We extend this test-vector counting method to account for path faults. For such a fault an entire path rather than a single stuck-at signal is sensitized. For our purpose, we consider sets of paths associated with each ELW interval, rather than single paths. Therefore, we associate an *interval ODC-mask $ODC(f,i)$* with each interval i in an $ELW(f)$. The one count of the interval ODC-mask is the interval weight. We proceed in reverse topological order by initially considering gates that feed primary outputs. For such gates all ODC-masks for intervals are simply equal to their ODC-masks (all 1's). For subsequent gates, ELWs are computed by translating and merging the ELWs of sink gates. Here each interval ODC associated with a sink gate is masked by the ODC of the gate in question to form the interval ODC for the current gate. Intuitively, the interval ODC mask keeps track of the observability of a signal through specific paths. Therefore masking the ODC corresponding to a path by the ODC of the additional gate simply adds that gate to the path.

When intervals from two fanout cones are merged, the interval ODC masks are unioned together using the bitwise OR operation. The OR operation results in some lack of accuracy for the weighting algorithm because it averages the weight for both intervals in the merged interval. However, this operation is necessary for scalability since each gate can be subject to exponentially many intervals and the loss of accuracy is small.

Suppose that a gate f has fanouts g and h and that during ELW computation, intervals $ELW(g)[i]$ and $ELW(h)[j]$ are merged together to form $ELW(f)[k]$. In this case,

$$ODC(f,k) = (ODC(g,i) + ODC(h,j))\&ODC(f)$$

The SER computation from interval weights is simply the sum of testabilities corresponding to each interval, weighted by the length of the interval in question. The testabilities are in turn derived using the interval ODC and signal probabilities. These computations are shown below.

$$test1(f,i) = \text{ones}\big(sig(f)\&ODC(f,i)\big)/K$$

$$Tmask(f,i) = (E_{fi} - S_{fi})/C$$

$$SER(C) = \sum_{f\in C}\sum_{i\in ELW(f)} \big(test1(f,i)gerr0(f) + test0(f,i)gerr1(f)\big)Tmask(f,i) \tag{2}$$

4.3 SER Evaluation Framework

We incorporate our timing masking analysis into AnSER, the reliability evaluation tool from [9] by combining logic and timing masking as shown in Equations 1 and 2. We use the bit-parallel logic simulations implemented in AnSER to compute signatures and ODC masks for the static and statistical algorithms. Along with timing masking, AnSER forms a scalable, lightweight method for guiding logic and physical synthesis flows towards increased reliability. It can also be used to simply check the impact of logic and physical synthesis techniques on reliability, possibly rejecting logic transformations or physical relocations whose impact on reliability is unacceptable. The scalability is in large part due to the efficient linear time algorithms that are used to compute the impact of logic and timing masking.

To capture electrical masking in AnSER, we can derating every gate error probability ($gerr0, gerr1$) by a factor dependent upon characterizations of sink gates. Researchers have shown that electrical masking eliminates weak glitches in 3-4 levels of logic, and has little effect thereafter [16]. This implies that considering paths of length 4 starting from the gate in question is sufficient. Further, the impact of electrical masking is expected to diminish with voltage scaling. AnSER is designed to guide CAD tools, where logic and timing masking are the primary mechanisms by which reliability can be increased.

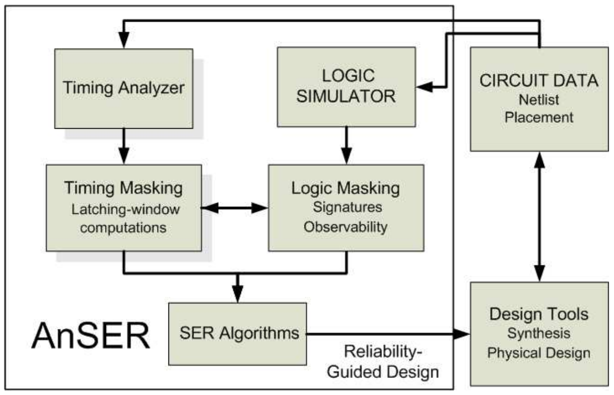

Figure 5: SER evaluation framework including logic and timing masking.

Figure 5 shows how to incorporate our method into a typical design flow RTL-to-GDSII. After each change to the netlist or placement, AnSER can be invoked incrementally. Physical changes only require the ELWs and signatures of fanin cones to be updated; logic changes can require both input and output cone signatures and ODCs to be updated. Since reliability is expressed as a summation in both cases, incremental evaluation involves regenerating signatures and ELWs for each gate in question. Unlike other reliability evaluators which often require a lot of circuit information, the amount of information processed and output by AnSER can be adjusted according to the needs of the user. For instance, if the user wishes to study the impact on reliability of only the timing optimizations steps, then she can use only timing computations in static mode. If a designer is only looking at logic transformations, then the logic-only mode may be used. Therefore the amount of coupling between the masking mechanisms can also be adjusted. AnSER can be connected to any external timing engine, and not necessarily the one used in our work.

5 Reliable Circuit Design

We now demonstrate the use of our reliability evaluator to guide two synthesis techniques. The first technique selects susceptible gates for radiation hardening by gate sizing and first was proposed by [23]. The second method uses gate relocation to reduce ELWs and improve reliability.

5.1 Gate Hardening

Gates are usually hardened by optimizing their width-to-length ratio in order to raise the energy threshold for particle strikes to result in errors. While gate hardening can be applied anywhere in the circuit, since it incurs area overhead, it is important to select specific critical gates to harden. We guide this technique using reliability evaluation rather than fault simulations (as in [23]) which can be computationally expensive and inaccurate if too few faults are sampled. Other papers that reported results on gate selection [11, 21] do not consider timing masking.

We select gates by considering their contribution to the overall circuit SER. This contribution can be estimated as the term corresponding to the gate in Equation 1.

$$sus(f) = \big(test1(g)gerr0(g) + test0(g)gerr1(g)\big)tmask(g) \tag{3}$$

This susceptibility metric takes into account several dependencies implicitly. Logic masking can overestimate the susceptibility of a gate because very observable gates can have smaller latching windows depending on the delay characteristics of a circuit. Equation 3 can be used as a pruning heuristic for candidate selection. Results show that while gate hardening can drastically improve reliability, different ways of guiding gate hardening can have drastically different results. For instance, guiding gate hardening by only logic masking models leads to different gates being selected then our combined logic-timing model.

Generally, gates are thought to have no susceptibility to errors after hardening [12]. Further, the impact of gate hardening on delay and area can be minimal if we take advantage of the fact that a small portion of gates are responsible for the majority of the SER. Since gate sizing can impact the delay of a gate, the ELW of gates in the fan-in cone of a sized gate could theoretically be altered. Therefore in order to perform gate hardening we pick one critical gate at a time, size it up, and then recompute the ELW in its fan-in cone. Since ELW calculation is fast, this can be done efficiently in our framework.

However, we note that ELW recomputation can be avoided in practice or batched to save runtime. Usually, when a transistor is sized up the propagation delay of the gate in question decreases. However, the gate itself presents a larger capacitive load to its driver and thus increases the propagation delay of the driving gate. However, the resized width can be adjusted such that the net impact on critical path delay is negligible because of the canceling effects. Zhou et al. [12] incur only 3% increase in delay when they size up 50% of gates.

5.2 Gate Relocation

As shown in Figure 2, gates with many different-length paths to outputs have the largest latching windows due to uneven path delay. Therefore, timing masking can be improved if some fanout paths are eliminated, or if the paths were made to have equal delay. However, unlike timing optimizations which result in same-delay paths through the circuit, our aim is to have equal delays to outputs *from each gate* in the circuit.

Embedding the reliability evaluator within a placement tool or closely coupling a placement algorithm with reliability goals is one way of tackling this problem. However, we take a lighter approach by making local changes to pre-placed designs. Specifically, we locally relocate nodes within the bounding box defined by their adjacent gates. Global characteristics of the placement are maintained in this way.

If a gate f has two fanouts g and h then the ELW of f can be translated by adding or subtracting delay from the g-to-f path and g-to-h path in such a way that the overlap is maximized when ELW(g) and ELW(h) are merged to form ELW(f). Since each ELW consists of a set of intervals, separated by various distances, computing a new (x,y) position for the gate such that the respective delays of the g-to-f and g-to-f yield a small latching window is a non-linear constrained optimization problem even for the local move of a single gate. We conjecture that the best location is likely to be near the center-of-gravity of the sources and sinks of the gate, and try legal neighboring locations as well. We move in reverse-topological order because the latching windows of gates near primary outputs affect the latching windows of earlier gates but not vice versa. Our results suggest that these gate relocations can improve reliability while maintaining delay. We expect this technique to have greater impact when interconnect delay forms a large portion of circuit delay.

Circuit	Gates	Runtime (s)			
		AnSER	SERD[16]	FASER [21]	[5]
c432	246	**<0.01**	10	22	—
c880	591	**<0.01**	10	—	—
c1355	746	**0.014**	20	40	2.09
c1908	760	**0.015**	20	66	0.781
c3540	1951	**<0.01**	60	149	5m42s
c6280	4836	**1.00**	120	278	—

Table 2: Comparison of SER evaluators.

6 Empirical Analysis

We now report some empirical results for error latching window computation and reliability improvement. The experiments were conducted on a 2.4 GHz AMD Athlon 4000+ workstation with 2GB of RAM. The algorithms were implemented in C++. We evaluated our algorithms on circuits from the IWLS 2005 benchmark suite [8], with design utilization set to 70% to match recent practice in industry. Our wire and gate characterizations are based on a 65nm technology library. We perform static timing analysis using the D2M delay metric [2] on Rectilinear Steiner Minimal Trees (RSMTs) produced by FLUTE [6]; Our designs placed using Capo version 10.2 [4, 20] and relocations are legalized using the legalizer provided by GSRC Bookshelf [20].

Table 1 shows changes in SER when timing masking is considered according to Equation 1. Base gate error probabilities (gerr0, gerr1) were derived using SPICE gate characterizations where a NAND gate has error rate $4e-5$ FIT [16]. SER incorporating timing masking can be useful in guiding physical synthesis operations while only considering logic masking is sufficient for technology-independent logic synthesis steps in the design flow. Table 1 also shows the potential for improvement in timing masking, i.e., the improvement in reliability when the ELW of each gate is made as small as possible (equal to the ELW of a latch). This shows that SER can be significantly decreased by manipulating timing masking.

Table 2 shows runtime comparisons of the extended AnSER tool (Figure 5) with other evaluators on the ISCAS 85 benchmarks. As seen in Table 2 and consistent with [9], AnSER runs orders of magnitude faster than other evaluators due to the linear-time algorithms for logic and timing masking. The logic masking model in AnSER was validated with the ATALANTA ATPG tool with high accuracy ($<3\%$ error). Other algorithms that explicitly account for logic masking tend to be much slower due to the difficulty of capturing the dependence on input vectors. BDD-based approaches like FASER [21] and MARS-C [11] tend to run out of memory on larger circuits due to input space explosion.

Table 3 shows improvements achieved by guiding gate hardening. Hardening the top 10% of the most susceptible gates leads to an average of 43% decrease in SER. Gates were selected using the susceptibility from Equation 3. The first column of this table shows the percentage of most susceptible gates that were not identified using logic masking alone. This indicates that guiding hardening with a timing masking model leads to different gates being hardened.

Table 4 shows the results of locally relocating gates

Circuit	No. gates	Clock period (secs)	Logic SER (FIT)	Runtime(s) (secs)	Timing SER (FIT)	Runtime (secs)	Potential % improvement
aes_core	20265	5.68E-07	0.1654	6	9.33E-05	3	37.57
spi	2998	3.19E-07	0.05722	1	4.23E-05	1	15.28
s35932	5545	6.18E-07	0.1363	2	6.03E-05	1	26.73
s38417	6714	3.56E-07	0.1360	2	1.22E-04	1	37.83
tv80	6802	6.79E-07	0.05602	2	2.64E-05	1	37.50
mem_ctrl	11062	6.44E-07	0.2185	2	8.45E-05	3	19.64
ethernet	36227	1.46E-06	0.7010	9	1.31E-04	9	91.68
usb_funct	10357	5.06E-07	0.1852	3	8.79E-4	3	36.59

Table 1: SER evaluation logic and timing masking.

Circuit	% New critical gates	SER (FIT)	% Decrease SER
aes_core	21.86	5.57E-05	40.29
spi	53.51	3.15E-05	25.43
s35932	57.03	3.80E-05	36.92
s38417	87.63	7.34E-05	40.30
tv80	33.67	1.39E-05	47.42
mem_ctrl	64.54	5.80E-05	31.36
ethernet	83.51	8.28E-05	36.67
usb_funct	88.96	8.70E-05	90.11
Average	61.34		43.56

Table 3: SER improvements through gate hardening.

	$65nm$		$< 45nm$	
Circuit	%SER	% delay	%SER	% delay
aes_core	11.83	3.00	21.15	-3.10
spi	18.87	4.8	41.62	-2.90
s35932	10.74	-0.13	44.02	3.40
s38417	10.10	1.38	14.35	-11.57
tv80	4.89	1.45	43.62	17.50
mem_ctrl	7.75	1.14	78.43	-1.70
ethernet	19.07	0.43	75.17	6.04
usb_funct	28.50	-5.26	14.29	-9.09
Average	13.97	0.55	41.59	2.10

Table 4: Improvements in SER through gate relocation.

within the bounding box of adjacent gates. We only accept changes that affect delay and SER positively, however legalization can later increase delay slightly. Our results indicate a 14% improvement at the 65nm technology node where average intrinsic gate delay is approximately a factor of $100x$ larger than (unit) interconnect RC delay. The second two columns project to smaller technology nodes where wire delay is expected to become comparable to gate delay. Such trends are indicated in the ITRS 2005 report on interconnects, which projects that at 32nm, wiring will contribute 90% of the circuit delay. The first set of results indicate a 14% decrease in SER while the second set shows a 41.59% decrease. Therefore, as technology scales timing masking can offer greater potential for improvement in SER.

7 Conclusions

We have proposed new algorithms for the static analysis of timing masking in SER evaluation. We also incorporated timing masking computations into a comprehensive signature-based SER estimation framework. We demonstrated latching window computation for selecting critical gates to harden. Our data shows a 43% decrease in SER when 10% of the most critical gates are hardened. Further, we presented a gate relocation technique for reducing latching window sizes for gates with multiple paths to outputs. Results show a 14% improvement in SER for 65nm and a projected improvement of 40% in technology where gate delay and interconnect unit RC delay become comparable. We conclude that timing masking can be utilized to achieve greater reliability with low area and delay overhead.

Acknowledgement: This research was sponsored in part by the Air Force Research Laboratory under Agreement No. FA8750-05-1-0282 and by the National Science Foundation under Grant CCF-0702276.

References

[1] S. Almukhaizim et al., "Seamless Integration of SER in Rewiring-Based Design Space Exploration," *ITC 2006*, pp. 1-9.

[2] C. J. Alpert, A. Devgan, C. Kashyap, "A Two Moment RC Delay Metric for Performance Optimization," *ISPD 2000*, pp. 69-74.

[3] C. J. Alpert, et al., "Techniques for Fast Physical Synthesis," *IEEE*, March 2007, vol. 95, no. 3, pp. 573-599.

[4] A. Caldwell, A. Kahng, and I. Markov, "Can Recursive Bisection Alone Produce Routable Placements?", *DAC 2000*, pp. 693-698.

[5] M. Choudhury, K. Mohanram, "Accurate and Scalable Reliability Analysis of Logic Circuits," *DATE 2007*, pp. 1454-1459.

[6] C. Chu, Y. -C. Wong, "Fast and Accurate Rectilinear Steiner Minimal Tree Algorithm for VLSI Design," *ISPD 2005*, pp. 28-35.

[7] D. Ernst et al., "Razor:: Circuit-Level Correction of Timing Errors for Low Power Operation," *IEEE Micro*, vol. 24, no. 6, Nov.-Dec 2003, pp. 10-20.

[8] IWLS 05 Benchmarks
http://iwls.org/iwls2005/benchmarks.html

[9] S. Krishnaswamy, S. M. Plaza, I. L. Markov, J.P. Hayes, "Enhancing Design Robustness with Reliability-aware Resynthesis and Logic Simulation," *ICCAD 2007* pp. 149-154.

[10] H. K. Lee, D. S. Ha, "On the Generation of Test Patterns for Combinational Circuits," TR No. 12-93, EE Dept., Virginia Polytechnic Institute.

[11] N. Miskov-Zivanov, D. Marculescu, "MARS-C: Modeling and Reduction of Soft Errors in Combinational Circuits," *DAC 2006*, pp.767-772.

[12] K. Mohanram, N. A. Touba, "Partial Error Masking to Reduce Soft Error Failure Rate in Logic Circuits" *DFT 2003*, pp. 433-440.

[13] M. Nicolaidis, "Time Redundancy Based Soft-Error Tolerant Circuits to Rescue Very Deep Submicron'," *VTS 1999*, pp. 86-94.

[14] S. Plaza, K-H. Chang, I. Markov, V. Bertacco, "Node Mergers in the Presence of Don't Cares" *ASP-DAC 2007*, pp. 414-419.

[15] R. Rao, D. Blaauw, D. Sylvester, "Soft Error Reduction in Combinational Logic Using Gate Resizing and Flipflop Selection," *ICCAD 2006*, pp. 502-509.

[16] R. Rao, et al., "An Efficient Static Algorithm for Computing the Soft Error Rates of Combinational Circuits," *DATE 2006*, pp. 164-169.

[17] P. Shivakumar, et al., "Modeling the Effect of Technology Trends on Soft Error Rate of Combinational Logic" *DSN 2002*, pp. 389-398.

[18] J. G. Tryon, "Quadded Logic," *Redundancy Techniques for Computing Systems*, 1962, pp. 205-228.

[19] J. von Neumann,"Probabilistic Logics & Synthesis of Reliable Organisms from Unreliable Components," *Automata Studies*, 1956.

[20] UMICH Physical Design Tools
http://vlsicad.eecs.umich.edu/BK/PDtools/

[21] B. Zhang, W. S. Wang, M. Orshansky, "FASER: Fast Analysis of Soft Error Susceptibility for Cell-Based Designs," *ISQED 2006*, pp. 755-760.

[22] M. Zhang, N. R. Shanbhag, "A Soft Error Rate Analysis (SERA) Methodology," *ICCAD 2004*, pp. 111-118.

[23] Q. Zhou, K. Mohanram, "Gate Sizing to Radiation Harden Combinational Logic," *TCAD*, vol. 25, no. 1, January 2006, pp. 155-166.