Improving Testability and Soft-Error Resilience through Retiming [*]

Smita Krishnaswamy [†], Igor L. Markov [♯], John P. Hayes [♯]
[†] IBM T.J. Watson Research Center, Rt. 134, Yorktown Heights, NY 10598
[♯] University of Michigan, EECS Department, Ann Arbor, MI 41809
{smita, imarkov, jhayes}@eecs.umich.edu

Abstract

State elements are increasingly vulnerable to soft errors due to their decreasing size, and the fact that latched errors cannot be completely eliminated by electrical or timing masking. Most prior methods of reducing the soft-error rate (SER) involve combinational redesign, which tends to add area and decrease testability, the latter a concern due to the prevalence of manufacturing defects. Our work explores the fundamental relations between the SER of sequential circuits and their testability in scan mode, and appears to be the first to improve both through retiming. Our retiming methodology relocates registers so that 1) registers become less observable with respect to primary outputs, thereby decreasing overall SER, and 2) combinational nodes become more observable with respect to registers (but not with respect to primary outputs), thereby increasing scan-testability. We present experimental results which show an average decrease of 42% in the SER of latches, and an average improvement of 31% random-pattern testability.

Categories and Subject Descriptors B.6.2 [**Logic Design**] Reliability and Testing— *Redundant Design, Testability*
General Terms Algorithms, Design, Reliability
Keywords Testability, Soft Errors, Retiming

1 Introduction

Single-event upsets (SEUs) caused by α-particles, high-energy neutrons, and cosmic rays are of concern in CMOS logic circuits. As the energy threshold for causing an error decreases, the number of particles with sufficient energy increases rapidly [18]. For instance, at lower energy thresholds, even trace amounts of radioactive contaminants in solder can affect CMOS circuits [8]. Additionally, transient errors in ICs may occur through a variety of hard-to-model phenomena, including capacitive and inductive noise, as well as thermal and power supply fluctuations.

At the same time, circuit test is becoming more important because fabrication process parameters are harder to control in sub-wavelength lithography. Fluctuations in dopant concentrations, transistor gate length, wire shapes, and via alignments can lead to hard errors in chips [2]. Therefore, methods that increase testability without compromising SER are necessary to identify incorrectly manufactured chips.

Heidel et al. [8] observe that registers are a major contributor to fail rates in high-performance ICs since latched errors are often not subject to electrical or timing masking. Several papers propose partial replication to improve the reliability of combinational logic [11, 15]. However, these methods only account for the combinational portion of the logic circuit and often incur significant area overhead. Further, these methods have to be used sparingly in order to maintain testability. Other techniques [21, 16] utilize electrical and timing masking to prevent the latching of errors originating in combinational logic. These techniques do not affect previously latched errors.

Errors in combinational logic are only becoming problematic now, while errors in registers are already a problem for critical applications [8]. In this paper, we specifically aim to reduce the soft-error susceptibility of registers, while simultaneously improving circuit testability. The main idea is to design circuits such that, even if a register experiences an SEU, its chances of propagating to a primary output are small. We account for logic masking in both combinational logic and registers during sequential operation and use this information to improve both the overall SER and testability of the design through retiming. Since we focus on logic masking, our solution is applicable to the various sources of errors mentioned above.

Retiming is the process of relocating registers to improve an objective (usually area or clock period) such that the functionality of the circuit remains unchanged.[1] Our retiming method utilizes the relationship between signal observability, soft-error propagation, and random-pattern testability. We derive linear programs (LPs) that relocate registers so as to minimize their sequential observability. Our main contributions are:

- An observability-based method for computing the error-susceptibility, and random-pattern testability for potential register locations in a sequential circuit.
- Linear programs for retiming that reduces circuit vulnerability to soft errors, and improves circuit testability simultaneously.

The remainder of the paper is organized as follows. Section 2 reviews relevant previous work on register retiming and functional simulation. Section 3 analyzes the effects of register relocation on error propagation and testability. Section 4 presents our retiming formulation and various extensions. Empirical validation is given in Section 5, followed by conclusions in Section 6.

2 Background

We now summarize the necessary background in retiming, soft-error mitigation, and testability.

2.1 Previous Work on Retiming

Leiserson and Saxe [10] first developed algorithms for minimum-period and -area retiming of edge-triggered circuits. For minimum-area retiming, a sequential circuit is represented by a graph $G(V,E)$, where each vertex $v \in V$ represents a combinational gate, and each edge $(u,v) \in E$

[1]In the past, the verification of retiming has been a problem, but solutions available in the past 4-5 years facilitate the practical use of retiming [14].

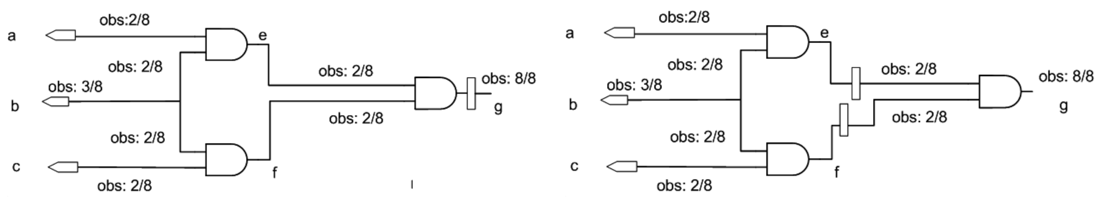

Figure 1: An illustration of observability reduction through retiming.

represents a connection between a driver u and sink v. An edge is labeled by a weight $w(u,v)$, indicating the number of registers (flip-flops) between u and v. The objective of minimum-area retiming is to determine labels $r(v)$ for each vertex v such that the total sum of edge weights is minimized. Here, $r(v)$ denotes the number of registers that are moved from the outputs to the inputs of v. The weight of an edge after retiming is given by:

$$w_r(u,v) = w(u,v) - r(u) + r(v)$$

Therefore, the total number of registers in the retimed circuit can be minimized by the following expression.

$$\sum_{(u,v)\in E} w(u,v) - r(u) + r(v)$$

Additionally, the retiming labels have to meet *legality* constraints, $w(u,v) \geq r(u) - r(v)$, for each edge to enforce the fact that edges cannot have negative weights. A linear program for the minimum-area retiming problem is given in Figure 2. Leiserson and Saxe [10] observe that this problem is the dual of a min-cost network flow problem and can therefore be solved in polynomial time.

```
 Minimize
    $\Sigma_{(\mathbf{u},\mathbf{v})\in\mathbf{E}} \mathbf{w}(\mathbf{u},\mathbf{v}) - \mathbf{r}(\mathbf{u}) + \mathbf{r}(\mathbf{v})$

subject to
 $\forall(\mathbf{u},\mathbf{v}) \in \mathbf{E}, \mathbf{r}(\mathbf{u}) - \mathbf{r}(\mathbf{v}) \leq \mathbf{w}(\mathbf{u},\mathbf{v})$
```

Figure 2: An LP for minimum-area retiming.

It is also possible to constrain the period in minimum-area retiming by ensuring that every path between two vertices with greater delay than the target period P has weight ≥ 1. In minimum-period retiming a binary search is conducted for the target clock period P and the feasibility of each period according to the legality constraints is checked using the Bellman-Ford algorithm [10].

Aspects of circuit testing have also been improved using retiming. Dey and Chakradhar [5] aim to reduce the lengths of partial scan chains in order to decrease testing time. Das and Bhattacharya [4] observe that combinational redundancies can be converted into sequential redundancies (unobservable changes in the state diagram) to improve the scan-based testability of circuits. In [20], the authors use the reverse process to convert sequential redundancies to combinational ones and then remove this redundancy using combinational optimization techniques. We note that these works are significantly different in their focus from ours. We aim to reduce the average observability – in effect the random-pattern observability of registers during normal operation. However, given that many registers are scanned, this retiming improves their observability during testing.

2.2 Techniques for SER Mitigation

Several techniques have been developed for improving the SER of logic circuits. These can be categorized by their error-mitigation mechanism. Soft errors in combinational logic are affected by three sources of masking [19]: 1) logic masking, where errors stop propagation due to the lack of a sensitized path to primary outputs or latches; 2) electrical masking, where soft errors are attenuated before being latched because of insufficient glitch duration or amplitude; 3) timing masking, when soft errors arrive at a register prior to a latching clock edge. Of these, only logic masking affects latched soft errors since latches can retain and drive erroneous values. Soft errors in combinational logic, however, are merely glitches that are likely to disappear due to the above mentioned sources of masking.

Techniques that increase logic masking include triple-modular redundancy, partial logic replication [15], guided rewiring [1], and signature-based partial redundancy addition [11]. These techniques mask errors on state elements, but often adversely affect testability and generally involve significant area overhead. Error-correction techniques such as Reed-Solomon and Hamming codes can also directly be used for state encoding, but these techniques incur so much overhead that they are not considered practical.

Techniques such as BISER [21, 16] use repeated sampling to determine the value of a signal before latching, relying on the assumption that erroneous glitches have shorter duration than the difference between sampling times. This assumption does not hold for errors already present in latches. Gate hardening [3] increases the energy threshold for error propagation such that only high-energy particles cause soft errors. While gate hardening can be used for state elements, this technique becomes less effective and requires proportionally more overhead as device technologies shrink. In this paper, we specifically focus on mitigating errors in registers through logic masking.

2.3 Logic Masking and Testability

A soft error in a logic circuit is propagated to a primary output only if there is a sensitized and observable path from the error to the output. This occurs only when a suitable test vector is applied at the primary input. Therefore, estimating the fraction of test vectors or *testability* is equivalent to estimating the probability that the error propagates to at least one primary output. This testability measure can be computed through functional simulation signatures in linear time. Note that this is testability with respect to the primary outputs. An alternative testability measure can be computed for scan-mode, i.e., assuming that the latches are scanned. We utilize these differing notions of testability—one applicable in sequential operation, and one applicable in scan-mode—to improve reliability and testability.

Two major parameters affecting testability are signal probability (controllability) and observability. Here, we review the use of functional simulation signatures for estimating the sequential observability of nodes in a circuit in linear time. Signal probability is computed in [11] by simulating random input vectors through logic in topological order. The collection of output responses at logic gates with respect to a collection 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 with respect to the input vectors is given by $Sig(f) = \{f(v_1), f(v_2), f(v_3), \ldots f(v_k)\}$.

To evaluate signal observability, observability don't-care masks (ODC masks) are computed from the signatures [17]. These correspond to input vectors for which the value of the signal affects a primary output. Observability is computed in reverse topological order by flipping bits in the signature and checking if the change is propagated to a primary output. For greater efficiency, this computation can be done in two steps. The first step is to check whether the change locally propagates through neighboring gates. The second step is to check for further propagation through the circuit by examining pre-computed ODC masks of the outputs of the neighboring gates. Corresponding to the K-bit signature $sig(f)$, we define $ODC(f)$ as the K-bit sequence whose i-th bit is 0 if input vector X_i is in the don't-care set of f; otherwise the ith bit is 1. Formally, $ODC(g) = \big(X_1 \in care(F_g), X_2 \in care(F_g), \ldots, X_K \in care(F_g)\big)$. Figure 3 shows an example of signature and ODC mask computation on a small circuit. Figure 4 summarizes the ODC computation algorithm of [17] for reference.

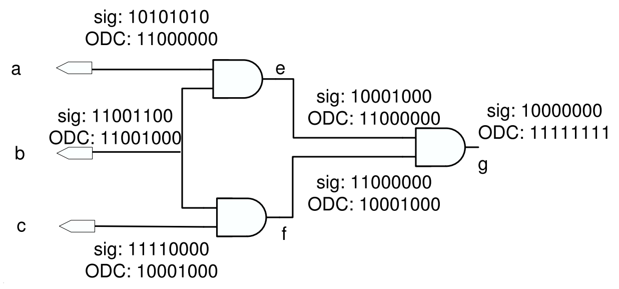

Figure 3: Signature and ODC illustration.

```
compute_odc_approx(Circuit C, size K)
{
  compute_sigs(C,K)
  sort_reverse_topological(C)
  for(all nodes $g \in C$)
    newsig(g) = sig(g)
    for(each fan-out branch $f \in$ fanout(g))
      sig(f) = Op<f>(inputsigs(f))
      localodc(g,f) = newsig(f) $\oplus$ sig(f)
      globalodc(g,f) = localodc(g,f)&odc(f)
      odc(g)| = globalodc(g,f)
}
```

Figure 4: An approximate ODC computation algorithm.

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

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

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

This measure for testability was validated by comparison to ATPG software in [11]. The SER, incorporating logic masking, is simply the sum of testabilities at each node f, weighted by gate error rates $gerr0(f)$ and $gerr1(f)$ (given in units of FITs) for 1-to-0 and 0-to-1 errors respectively.

$$SER(C) = \sum_{f \in C} test_1(f)gerr0(f) + test_1(f)gerr1(f)$$

The logical SER is generally proportional to the observability, and only different by a multiplicative factor if the probabilities of 0-to-1 and 1-to-0 errors are the same. This assumption can be made due to lack of information during technology-independent optimization.

$$obs(f) = \text{num\_ones}\big(ODC(f)\big)/K$$

$$SER(C) = \sum_{f \in C} obs(f)gerr(f)$$

Note that logic design cannot change the actual probabilities of error occurrence ($gerr(0)$ and $gerr(1)$). However, design decisions can decrease the testability of registers or signals. Since we are performing technology-independent logic optimization for reliability, and do not have information about the relative probabilities of 0-to-1 vs 1-to-0 errors, we use the observability to improve the estimated SER.

3 Circuit Analysis

In this section, we analyze the effect of register relocation on the SER and testability of sequential circuits.

3.1 Retiming and Sequential SER

Retiming can improve circuit reliability by relocating registers such that soft errors are more likely to be masked. In order to account for error propagation through multiple stages, we modify the circuit using time-frame expansion. In an n-frame expansion, n copies of the circuit are made and each register is simply replaced by a wire. The outputs of the kth stage are fed back into the inputs of the $k+1$th stage (as appropriate). Register inputs in the 0th frame are treated as primary inputs and register outputs of the n frame are treated as primary outputs.

The sequential observability is similar to the measure introduced in the previous section. However, we consider multiple cycles of operation through time-frame expansion. Only the primary outputs of each frame are considered completely observable. This accounts for errors that propagate past a single cycle before appearing at a primary output. It is true that some errors in registers at the n-th frame may appear at primary outputs, but this is unlikely since the probability of errors in the n-th frame are small. Experiments reported in several publications [6, 13] show that the majority of errors are flushed out in 2-3 cycles.

Additionally, in order to compute observability from a reasonable set of start states we obtain a sample of reachable states by simulating the sequential circuit for 20 cycles starting from a reset state. Experiments have shown that 10-15 cycles of simulation suffice to reach steady state on most ISCAS89 benchmarks [7, 13]. We denote this measure $seqobs(f,n)$ where f is the name of the signal, n is the number of frames of expansion. The sequential observability is given by the fraction of ones in the ODC mask of f in the n-frame expanded circuit.

$$seqobs(f,n) = \text{ones}\big(ODC(f,n)\big)/K$$

The SER of a sequential circuit is therefore:

$$SER(C,n) = \sum_{f \in C} gerr0(f)seqobs(f,n)$$

If we separate the contributions of the registers from the contributions of the combinational logic to the SER of the sequential circuit, then we obtain:

$$SER\_Comb(C,n) = \sum_{f \in Comb(C)} gerr0(f)seqobs(f,n)$$

$$SER\_Reg(C,n) = \sum_{r \in Reg(C)} gerr0(r)seqobs(r,n)$$

$$SER(C) = SER\_Reg(C,n) + SER\_Comb(C,n)$$

The $SER\_Comb(C,n)$ portion of the error does not change under register relocation, since functionally, registers simply transmit the input signal to the output unchanged (after some delay). Therefore, registers do not logically mask errors when considering multi-cycle operation, and in turn, do not affect the observability of other nodes in the circuit. However, $SER\_Reg(C,n)$, does change with the movement of registers. For instance, if registers move from the output of a node f to its inputs then, errors on the registers can be additionally logically masked by f. The sequential observability of the a register r is generally (with some exceptions) the same as that of its driving gate, therefore if a register is moved, its sequential observability changes. This suggests that registers should be placed in locations such that their sequential observability is low.

3.2 Retiming and Random-Pattern Testability

Since the signature-based framework computes the testability of a circuit based on random simulation vectors, $test_1(f)$ computes the random-pattern testability of node f for 0-1 errors, and $test_0(f)$ computes the same for 1-0 errors. In the absence of registers, the random-pattern testability of the entire circuit can be computed by:

$$Rand\_Test(C,n) = \frac{1}{|Comb(C)|} \sum_{f \in Comb(C)} test_1(f,n) + test_0(f,n)$$

In modern logic circuits, most registers are directly scanned out and read in test mode, i.e., registers can be treated as primary outputs when considering testability. Therefore, if registers were added to nodes f with low testability, then $Rand\_Test(C)$ would increase. This again leads to the conclusion that registers should be placed in regions of low observability.

The random-pattern testability of a circuit can be improved iteratively. At iteration 0, we assume that there are no fixed registers in the design. Therefore, $Rand\_Test(C,n)$ is improved by placing registers in locations of low observability—in our case, through retiming. In iteration 1, the testability is analyzed with respect to the current register locations. Additional test points are placed at locations that have low observability with respect to the circuit of iteration 0, and so on. Therefore, iteration 0 of this process involves the same goal as minimizing SER, i.e., decreasing the total sequential observability of registers.

Example 1 *For the circuit in Figure 1, the testabilities of nodes e, f and g are as follows.*
$test_0(e,1) = 1/8, test_1(e,0) = 1/8$
$test_0(f,1) = 1/8, test_1(e,0) = 1/8$
$test_0(g,1) = 7/8, test_1(g,0) = 1/8$
The random-pattern testability of this circuit is

$$Rand\_Test(C,n) = (2/8 + 2/8 + 1)(1/3) = (1/2)$$

In this circuit registers should be placed at nodes e and f due to their low observability.

4 Capturing Retiming by Linear Programs

We now derive LPs for retiming, accounting for the sequential observability of each register location. First, we present the basic retiming formulation assuming no register sharing, i.e, if a latch driven by a node u has fanouts v, w, then we model this as though there was a latch both at (u,v) and (u,w). In other words $w(u,v) = w(u,w) = 1$. Then, we account for register sharing at fanout branches.

4.1 Minimum-Observability Retiming

The sequential observability of each edge (u,v) is the same as the output of a buffer that is placed on edge (u,v). We denote the observability of edge (u,v), $seqobs((u,v),n)$ which is computed using signatures as described in Section 2. In the case where u only has one fanout $seqobs((u,v),n) = seqobs(u,n)$, in other words, the observability is the same as that of its driver. Since registers logically act as buffers, a register output has the same observability as a register input, and edges with registers still have the same observability after the registers are moved. The objective function accounting for total register observability is given by:

$$\sum_{(u,v)\in E} w_r(u,v)seqobs((u,v),n)$$

Additionally, if u is a primary input then $r(u)$ is necessarily 0 and similarly for v. This ensures that no peripheral retiming is done, and that the overall period of the circuit does not increase beyond the longest combinational path in the module being optimized. The modified LP is shown in Figure 5.

Minimize
$\sum_{(\mathbf{u},\mathbf{v})\in \mathbf{E}} (\mathbf{w}(\mathbf{u},\mathbf{v}) - \mathbf{r}(\mathbf{u}) + \mathbf{r}(\mathbf{v}))\mathbf{seqobs}((\mathbf{u},\mathbf{v}),\mathbf{n})$

subject to
$\forall (\mathbf{u},\mathbf{v}) \in \mathbf{E}, \mathbf{r}(\mathbf{u}) - \mathbf{r}(\mathbf{v}) \leq \mathbf{w}(\mathbf{u},\mathbf{v})$

Figure 5: Minimum-observability retiming formulation.

Example 2 *For the circuit shown in Figure 1 the edges include $(a,e),(b,e),(b,f),(c,f),(e,g),(f,g),(g,o)$. Note that input and output wires are also considered valid edges. However, we only derive retiming labels for the intermediate nodes e, f, g. The objective function is:*
$w_r(a,e)(2/8) + (w_r(b,e) + w_r(b,f))(3/8) + w_r(c,f)(w/8)$
$+ w_r(e,g)(2/8) + w_r(f,g)(2/8) + w_r(g,o)(8/8)$
The retimed weight, for instance, of edge (e,g) is $w_r(e,g) = w(e,g) - r(e) + r(g)$.

Once the circuit is retimed, registers can be shared again during post-processing. For instance, if edges (u,v) and (u,w) both have registers after retiming then these are simply shared. In general, the number of registers required at the output of u is $max(w_r(u,f_1), w_r(u,f_2) \ldots w_r(u,f_n))$where $f_1, f_2, \ldots f_n$ are fanouts of u.

The formulation in Figure 5 can be modified to constrain the area and period of the circuit. For area constraints, we can perform a binary search for the smallest feasible area M by including the constraint $(\sum_{(u,v)\in E} w(u,v) - r(u) + r(v)) < M$. The period can be constrained to a target P by the method of [10]. Here, the D matrix stores the delay of longest path between the vertices (u,v) in $D(u,v)$ and the W matrix stores the weight of the said path. These additional constraints are shown in Figure 6.

`Minimize`

$$\sum_{(\mathbf{u},\mathbf{v})\in\mathbf{E}}(\mathbf{w}(\mathbf{u},\mathbf{v})-\mathbf{r}(\mathbf{u})+\mathbf{r}(\mathbf{v}))\mathbf{seqobs}((\mathbf{u},\mathbf{v}),\mathbf{n})$$

`subject to`

$$\forall(\mathbf{u},\mathbf{v})\in\mathbf{E},\mathbf{r}(\mathbf{u})-\mathbf{r}(\mathbf{v})\leq\mathbf{w}(\mathbf{u},\mathbf{v})$$
$$(\sum_{(\mathbf{u},\mathbf{v})\in\mathbf{E}}\mathbf{w}(\mathbf{u},\mathbf{v})-\mathbf{r}(\mathbf{u})+\mathbf{r}(\mathbf{v}))<\mathbf{M}$$
$$\forall\mathbf{u},\mathbf{v}\in\mathbf{V}\quad\mathbf{s.t.}\quad\mathbf{D}(\mathbf{u},\mathbf{v})>\mathbf{P},\mathbf{r}(\mathbf{u})-\mathbf{r}(\mathbf{v})\leq\mathbf{W}(\mathbf{u},\mathbf{v})-\mathbf{1}$$

Figure 6: Area- and period- constrained retiming for minimum observability.

4.2 Incorporating Register Sharing

In the previous section, we derived a minimum-observability retiming formulation that does not account for register sharing. Hence, the optimization takes place on a version of the circuit with registers cloned at each fanout branch. The difficulty in incorporating sharing is that observability is a non-linear property of edges.

Example 3 *Consider again the circuit C of Figure 1. Suppose the retimed weights of edges* (b,e) *and* (b,f) *are* $w_r(b,e)=2$ *and* $w_r(b,f)=1$*. According to the formulation of Figure 5, the objective function for this portion of the circuit equals* $(2/8)w_r(b,e)+(2/8)w_r(e,g)=(1/3)$*. However, the two registers at* (b,f) *and* (b,e)*, can be replaced by a single register with fanouts to both e and f. This register does* not *have observability* $seqobs((b,e),n)+seqobs((b,e),n)=4/8$*. Instead, the observability is computed by counting the fraction of* $1's$ *in its ODC mask. The ODC mask, in turn, is computed as the bitwise* OR *of the ODC masks through each fanout, as shown in Figure 4. Thus,* $ODC(b)=globalODC(b,e)\mid globalODC(b,f)=3/8$.

For each register with driver u and fanouts $S=\{s_1,s_2\ldots s_m\}$, the correct sequential observability must be computed using the method of Figure 4. This observability is equivalent to the *seqobs* of a buffer with input u and outputs S, denoted $seqobs((u,S),n)$. Here, S is a subset of all of the fanout branches of u, $F_u=\{f_1,f_2,\ldots f_n\}$. For any node u with fanout branches $F_u=\{f_1,f_2,\ldots f_n\}$, we can compute the total number of registers that can be shared by any subset of these branches, using the edge weights introduced in the previous section as follows. The number of registers that can be shared by *all* the fanout branches is given by:

$$w_r(u,F_u)=min(w_r(u,f_1),w_r(u,f_2)\ldots w_r(u,f_n))$$

The number of registers shared by a subset of fanouts, $S=\{f_1,f_2,\ldots f_{n-1}\}\subset F_u$ of size $|F-u|-1$ is $min(w_r(u,f_1),w_r(u,f_2)\ldots w_r(u,f_{n-1}))-w_r(u,F_u)$. In general, the number of registers shared by a subset $S\subset F_u$ is the minimum weight of any edge of the form $(u,s_i),s_i\in S$, minus the registers that are shared by any larger subset S' of F_u. Hence, the total weight of a subset of fanout branches S, using the principle of inclusion and exclusion, is given by:

$$w_r(u,S)=\sum_{S':S\subset S'\subseteq F}(-1)^{(|S'|-|S|)}min(w_r(u,s'_1),w_r(u,s'_2),\ldots),s'_i\in S'$$

Then, these register counts, $w_r(u,S)$, are weighted by their sequential observability $seqobs((u,S),n)$. The sum of such quantities over all possible subsets of F_u, gives us the correct total observability of registers driven by u, assuming maximal sharing. Maximal sharing is desired because a shared register always has observability less than or equal to that of its cloned registers combined.

$$totobs(u)=\sum_{S:S\subseteq F_u}w_r(u,S)*seqobs((u,S),n)\qquad(1)$$

The function $totobs(u)$ has to be linearized in order to be incorporated into the LP. This requires linearizing the *min* function—the only non-linear element of the *totobs* function. Generally, the function $min(a_1,a_2,\ldots a_n)$, for any real values a_i can be linearized by introducing a new variable MIN, along with the constraints $MIN\leq a_1,MIN\leq a_2,\ldots MIN\leq a_n$. Then, the objective function has to maximize the value of MIN as LP converges to a solution so that $MIN=min(a_1,a_2\ldots a_n)$.

We introduce a variable $MIN_{u,S}$ for each function $min(w_r(u,s_1),w_r(u,s_2),w_r(u,s_3)\ldots)$ in $totobs(u,F_u)$. The associated constraints are of the form $MIN_{u,S}\leq w_r(u,s_1)$, $MIN_{u,S}\leq w_r(u,s_2)$, etc. Finally, we append $-c(MIN_{u,S})$ to the end of the objective function for each variable $MIN_{u,S}$ introduced. Here, c is any sufficiently large constant, i.e., $\forall S,c>>\sum seqobs((u,S),n)$. Since the retiming linear program has a minimization objective, the additional terms ensure that the $MIN_{u,S}$ variables are set to their highest (correct value) when the objective is optimized. The remaining retiming variables will be optimized for low observability as before. This altered LP, incorporating register sharing, is given in Figure 7.

`Minimize`

$$\sum_{(\mathbf{u})\in\mathbf{V}}\mathbf{totobs}(\mathbf{u})-(\mathbf{c}\sum_{\mathbf{S}\subset\mathbf{F_u}}\mathbf{MIN_{u,S}})$$

`subject to`

$$\forall\mathbf{u}\in\mathbf{V},\mathbf{S}\in\mathbf{F_u},\forall(\mathbf{s_i}\in\mathbf{S})\mathbf{Min_{u,S}}\leq\mathbf{w_r}(\mathbf{u},\mathbf{s_1})$$
$$\forall(\mathbf{u},\mathbf{v})\in\mathbf{E},\mathbf{r}(\mathbf{u})-\mathbf{r}(\mathbf{v})\leq\mathbf{w}(\mathbf{u},\mathbf{v})$$

Figure 7: Minimum-observability retiming formulation with register sharing.

While the formulation in Figure 7 correctly captures register sharing, it can become intractable for nodes with many fanouts. By recollecting the coefficients next to the MIN variables, we can write the *totobs* function as follows:

$$totobs(u)=\sum_{S\subseteq F_u}C_{u,S}Min(u,S),$$

$$C_{u,S}=\sum_{S':S'\subset S\subseteq F_u}(-1)^{|S|-|S'|}seqobs(u,S',n)$$

From this formulation, it is clear that the number of additional terms generated in the objective function for each node u is on the order of $2^{|F_u|}$. However, many practical circuits have low maximum fanout, due to drive-strength limitations of available standard cells.

5 Empirical Validation

We now describe experiments to validate our proposed retiming formulation. Our signature and observability computations are implemented in C++, while the linear programs are solved using CPLEX v.10.1 [9]. Note that an integer optimal solution is guaranteed without explicitly enforcing integer constraints [10].

Figure 8 summarizes the propagation of errors through sequential circuits, as estimated by bit-parallel functional simulation [11] extended to sequential circuits. The figure indicates that most errors are apparent at the outputs in immediate cycles after their occurrence. The error probability in later cycles diminishes rapidly. Therefore, we compute sequential observability from a ten-frame expansion of the circuit.

Table 1 shows results on ISCAS-89 benchmark circuits with the minimum-observability retiming formulation where each edge (u,v) is weighted by a 10-frame sequential observability measure. We use the formulation shown in Fig-

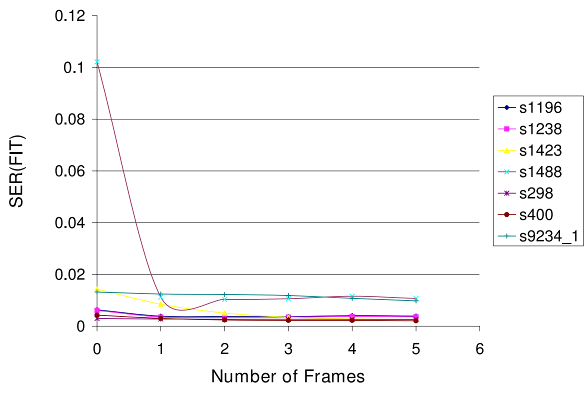

Figure 8: Error propagation in sequential circuits through multiple cycles of operation.

	Before		After		Change	
Circuit	No. FFs	Total Obs	No. FFs	Total Obs	% Area Inc.	% Obs. Dec.
s208	8	9.570	10	9.173	1.785	4.147
s298	14	15.579	34	5.059	15.037	67.527
s344	15	12.770	11	8.539	-2.29	33.126
s386	6	4.206	4	3.8256	-1.121	9.035
s444	21	14.752	34	4.146	6.435	71.895
s526	21	15.906	94	3.567	34.112	77.568
s832	5	25.311	30	2.669	8.561	89.3365
s1238	18	2.848	27	2.147	1.711	24.631
s1196	18	2.911	29	2.115	2.010	27.373
s1494	6	5.746	7	3.248	0.153	43.47
s1488	6	5.744	8	3.8256	0.303	42.787
s1423	73	25.273	115	20.115	5.753	20.110
Avg.					6.53	42.61

Table 1: Decrease in register observability through retiming.

Circuit	No. Gates	Scan Testability		
		Before	After	% Improved
s208	104	0.494	0.499	1.03
s298	119	0.523	0.699	33.72
s386	159	0.360	0.389	7.77
s444	181	0.474	0.689	45.42
s526	193	0.445	0.614	37.75
s832	287	0.222	0.368	65.28
s1238	508	0.300	0.317	5.78
s1196	529	0.302	0.321	6.45
s1494	647	0.269	0.413	53.69
s1488	653	0.267	0.410	53.57
s1423	657	0.424	0.552	30.08
Avg.				30.96

Table 2: Improvements in random-pattern scan-testability.

ure 5. The LP was solved in all cases in less than 0.1 seconds by *CPLEX*. The observability, which is proportional to the SER susceptibility is decreased by an average of 42% with only a 7% overall area increase. Table 2 shows an average improvement of 31% in the random-pattern testability of nodes in the combinational portion of the circuit. Our results indicate an interesting feature of error tolerance—that it is possible to decrease the overall sequential SER while increasing what is traditionally computed as the SER of a combinational logic circuit, i.e., with registers treated as primary outputs [12, 11]. Therefore, it is insufficient to solely consider combinational behavior for SER analysis. While combinational circuit optimization for SER remains important and generally decreases overall circuit SER, it is necessary to account for error propagation behavior at the sequential level. For instance, combinational circuits can be designed such that they reduce error propagation to registers with high observability.

6 Conclusions

We have developed a novel retiming approach to improve both the soft-error tolerance and testability of sequential circuits. The proposed techniques exploit certain relationships between observability, testability and SER, and incorporate them into linear programs for retiming optimization. We also extended these programs to handle area constraints and fanout sharing. Area-unconstrained minimization was shown to reduce the average error susceptibility of registers by 40%, and improve testability by 31% on average.

Acknowledgment. This work was sponsored in part by the Air Force Research Laboratory under Agreement No. FA8750-05-1-0282.

References

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

[2] S. Borkar, et al., "Parameter Variations and Impact on Circuits and Microarchitecture,"*DAC*, 2003, pp. 328-342.

[3] T. Calin, M. Nicolaidis, R. Velaco, "Upset Hardened Memory Design for Submicron CMOS Technology," *IEEE Trans. Nucl. Sci.*, Dec. 1996, vol. 43, pp. 2874-2878.

[4] D. K. Das, B.B. Bhattacharya, "Does Retiming Affect Redundancy in Sequential Circuits?," *VLSID*, 1996, pp. 260-263.

[5] S. Dey, S.T. Chakradhar,"Retiming Sequential Circuits to Enhance Testability," *VTS*, 1994, 25-28.

[6] J. P. Hayes, I. Polian, B. Becker, "An Analysis Framework for Transient-Error Tolerance," *VTS*, 2007, pp. 249-255.

[7] G.D. Hachtel, E. Macii, A. Pardo, F. Somenzi, "Markovian Analysis of Large Finite State Machines," *TCAD*, Dec. 1996, vol. 15, no. 12, pp. 1479-1493.

[8] D. F. Heidel et al., "Alpha-particle Induced Upsets in Advanced CMOS Circuits and Technology," *IBM Journal of Research and Dev.*, May 2008, vol. 52, no. 3, pp. 225-232.

[9] ILOG CPLEX: http://www.ilog.com/products/cplex

[10] C.E. Leiserson, J.B. Saxe,"Retiming Synchronous Circuitry," *Algorithmica*, 1991, vol. 6,pp. 5-35.

[11] 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.

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

[13] N. Miskov-Zivanov, D. Marculescu, "Modeling and Optimization for Soft-Error Reliability of Sequential Circuits," *IEEE TCAD*, 2008, vol 27. no.5 pp. 803-816.

[14] M. N. Mneimneh, K. A. Sakallah, "Principles of Sequential-Equivalence Verification," *IEEE DT*, 2005, vol. 22, no. 3, pp. 248-257.

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

[16] M. Nicolaidis,"Time Redundancy Based Soft-Error Tolerance to Rescue Nanometer Technologies," *VTS*, 1999, pp. 86-94.

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

[18] K. Rodbell, D. F. Heidel, et al., "Low-Energy Proton-Induced Single-Event-Upsets in 65 nm Node, Silicon-on-Insulator, Latches and Memory Cells," *IEEE trans. on Nuclear Science*, vol. 54, no. 6, Dec. 2007, pp. 2474-2479.

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

[20] H. Yotsuyanagi, S. Kajihara, K. Kinoshita, "Synthesis for Testability by Sequential Redundancy Removal using Retiming," *Fault-Tolerant Computing*, 1995, pp. 33-40.

[21] M. Zhang, S. Mitra, et al.,"Sequential Element Design with Built-in Soft Error-Resilience" *TVLSI,* Dec. 2006, vol. 14, no.12, pp. 1368-1378.