Enhancing Design Robustness with Reliability-aware Resynthesis and Logic Simulation

Smita Krishnaswamy, Stephen M. Plaza, Igor L. Markov, and John P. Hayes
{smita, splaza, imarkov, jhayes}@eecs.umich.edu
Advanced Computer Architecture Lab, University of Michigan
2260 Hayward Street, Ann Arbor 48109-2121, USA

Abstract

While circuit density and power efficiency increase with each major advance in IC technology, reliability with respect to soft errors tends to decrease. Current solutions to this problem such as TMR require high area and power overhead. In this work, soft-error reliability is improved with minimal area overhead by careful, localized circuit restructuring. The key idea is to increase logic masking of errors by taking advantage of conditions already present in the circuit, such as observability don't-cares. We describe two circuit modification techniques to improve reliability: covering-based resynthesis and local rewriting. A key feature of these techniques is fast, on-the-fly estimation of soft error rate (SER) using our reliability evaluator AnSER. This tool is compared against prior SER evaluators and found to run orders of magnitude faster. We show empirically that our reliability-driven synthesis methods can reduce SER by 29-40% with only 5-13% area overhead.

1 Introduction

Reliability with respect to soft (transient) errors is becoming an important concern in digital circuits, and ICs are now routinely characterized by their soft error rate (SER). To guide synthesis techniques toward more reliable circuits, fast methods are needed to identify beneficial design changes and re-evaluate SER after each change. While several SER evaluation tools have been developed recently [14, 15, 7, 11], they require layout and electrical information that is usually unavailable during synthesis at the technology-independent logical level. In this work, we describe reliability-directed, logic-level resynthesis techniques that employ a very fast SER evaluator called AnSER.

It has long been known that many soft failures are masked and do not lead to observable circuit errors. Hence, soft-error reliability can be improved by increasing masking opportunities, e.g., by logic replication such as in triple modular redundancy (TMR). However, these methods substantially increase a circuit's area and power consumption.

Our work demonstrates that logic masking can be enhanced systematically without significantly increasing circuit area. We accomplish this by local restructuring that takes advantage of inherent functional redundancy and don't-cares in the logic circuit. Figure 1 outlines our reliability-aware synthesis methodology, which exploits the intimate relations between logic masking, observability don't-cares (ODCs) at individual nodes of the circuit, functional simulation, and the testability of stuck-at faults. Bit-parallel logic simulation is used to compute signatures and don't-care masks for all signal nodes in a logic circuit. The signatures and don't-care masks serve as estimates of node

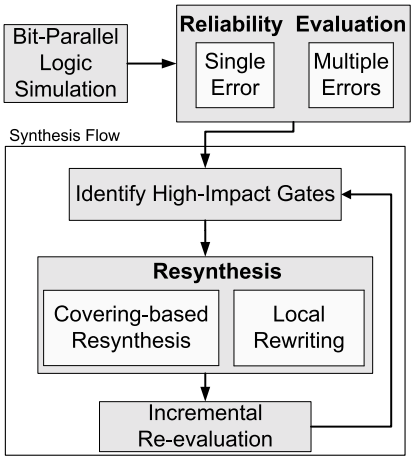

Figure 1: Proposed reliability-aware synthesis method.

observability, and facilitate a fast new way to identify regions of high impact with respect to reliability. We then resynthesize these *high-impact* regions using a reliability evaluator AnSER to simultaneously improve reliability and area. This paper's main contributions are:

- A fast incremental reliability evaluator AnSER that can be used stand-alone or integrated with logic synthesis
- A novel resynthesis technique to improve reliability by exploiting logical covering relationships and observability don't-cares
- Another resynthesis method based on local rewriting that improves reliability while decreasing area

The paper is organized as follows. Section 2 discusses previous work on reliability evaluation and reliability-driven synthesis. Section 3 covers background on bit-parallel simulation and signatures. Section 4 presents our reliability evaluation methodology. Section 5 describes two strategies for synthesis to improve reliability. Section 6 presents empirical results, while Section 7 concludes the paper.

2 Previous Work

Recent SER evaluators include SERA [15], FASER [14], MARS-C [7], and SERD [11]. They all perform gate pre-characterization using SPICE to estimate the probability with which a single-event upset (SEU) causes an erroneous glitch, as well as the probability that the glitch propagates through the circuit. Glitch propagation is governed by three masking mechanisms [12]: (i) logic masking (the glitch occurs in a non-sensitized portion of the circuit); (ii) electrical

masking (the glitch is attenuated and blocked by the electrical characteristics of CMOS gates); and (iii) temporal masking (the glitch occurs in a non-latching portion of the clock cycle). A new BDD-based SER evaluator in [4] offers advanced modeling of logic masking for reconvergent fanouts.

Several techniques are known to reduce the impact of soft errors. Rao et al. [10] use the algorithm from [11] to selectively resize gates and flip-flops. With gates of greater dimensions, low-energy particle strikes are less likely to cause a glitch. Larger gates also imply that glitches are less likely to appear at gate outputs, and those that do appear are more likely to be electrically masked.

Soft errors can also be mitigated by adding redundant logic. Classic techniques such as TMR and quadded logic [13] achieve this by systematically replicating logic. In quadded logic, each gate is replaced by a network of four gates which logically mask single errors. TMR triplicates the entire circuit and uses voters to mask errors. Mohanram and Touba [8] reduce the cost of TMR by replicating only the most susceptible gates. However, even partial replication of this kind is quite expensive.

More recently, Almukhaizim et al. [2] have proposed SER reduction via *rewiring* , i.e., logic transformations that reconnect wires without modifying gates. Their work appears to be the first example of reliability-guided circuit restructuring, but it has some limitations. Rewiring is much more restrictive than other synthesis techniques, such as *rewriting* [6], which can also add or remove gates. Rewiring also incorporates automatic test pattern generation (ATPG) which is often quite slow.

3 Signatures and ODC Masks

We make extensive use of *node signatures* to compute reliability, target high-impact areas of a circuit, and identify matching nodes for resynthesis. A circuit node g can be labeled by a *signature* $sig(g) = \big(F_g(X_1), F_g(X_2), \ldots, F_g(X_K)\big)$ defined as the sequence of logic values observed at g in response to a sequence of K input vectors $X_1, X_2, \ldots, X_K$. Here, $F_g(X_i) \in \{0,1\}$ indicates the value appearing at g in response to X_i. The signature $sig(g)$ thus partially specifies the Boolean function F_g realized by g. Applying all possible input vectors (exhaustive simulation) generates a signature that corresponds to a full truth table. Signatures with 64-1024 bits are useful in pruning non-equivalent nodes during equivalence-checking [16, 9].

Figure 2 shows a 5-input circuit where each of 10 nodes is labeled by an 8-bit signature *SIG* computed with eight given input vectors. Input vectors are often randomly sampled, and simulation propagates signatures to internal and output nodes. With bit-parallel simulation, signatures are manipulated with 64-bit logical operations, ensuring high simulation throughput. Generating K-bit signatures in an N-node circuit takes $O(NK)$ time.

Observability don't-cares (ODCs) occur at node g for certain input vectors when the values at g do not affect the primary outputs. For example, in the circuit $\mathrm{AND}(a, \mathrm{OR}(a,b))$, the output of the OR gate is inconsequential when $a = 0$. Corresponding to the K-bit signature $sig(g)$, we define $ODCmask(g)$ as the K-bit sequence whose i^{th} bit is 0 if input vector X_i is in the don't-care set of g; otherwise the i^{th} bit is 1. Formally, $ODCmask(g) = \big(X_1 \notin ODC(F_g), X_2 \notin ODC(F_g), \ldots, X_K \notin ODC(F_g)\big)$.

Bit-parallel simulation provides an efficient way to compute ODC masks, and we found the heuristic algorithm from [9] particularly convenient. This algorithm traverses the circuit in reverse topological order and, for each node, computes a local ODC mask for its immediate downstream gates. The local ODC masks are derived by flipping each value in the signature to see if the output of the gate changes. The local ODC masks are then bitwise-ANDed with the respective global ODC mask at the output of the gate in question to produce the global ODC mask of the node for paths through the particular fanout branch. The global ODC masks for *all* fanout branches are then ORed to produce the final ODC mask for the node. The ORing takes into account the fact that a node is observable for an input vector if it is observable along *any* of its fanout branches. Reconvergent fanout can eventually lead to incorrect values. However, the masks can be corrected by performing exact simulation downstream from the converging nodes.

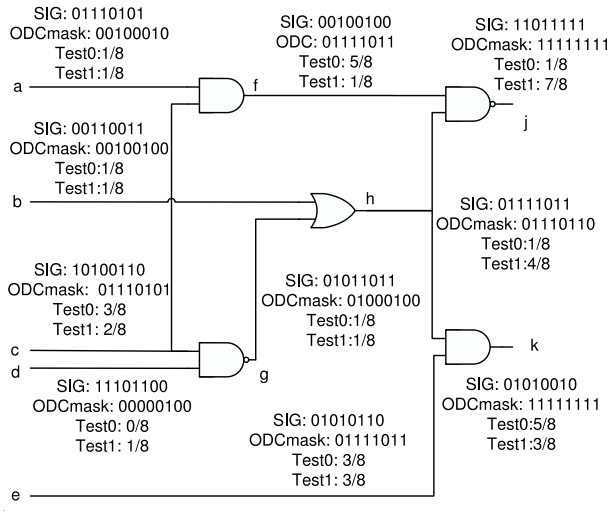

Figure 2: Signatures, ODC masks, and testability information associated with circuit nodes.

Example 1 *Figure 2 shows a sample* 8*-bit signature and the accompanying ODC mask for each node of a* 10*-node circuit. The ODC mask at c, for instance, is derived by computing ODC masks for paths through nodes f and g respectively and then ORing the two. The local ODC mask of c for the gate through f is* 01110101*. When this is ANDed with the ODC mask of f, we find the global ODC of c on paths through f,* 01110001*. Similarly, the local ODC mask of c for the gate with output g* 11101100*, and the global ODC mask for paths through g is* 01000100*. We get the ODC mask of c by ORing the ODC masks for paths through f and g, which yields* 01110101*.*

4 Reliability Evaluation

We now present the soft-error reliability evaluator AnSER, which was specifically designed for use in logic synthesis.

4.1 Fault Model for Soft Errors

Integrating reliability evaluation efficiently into logic synthesis requires scalability and logical fault models that are technology independent. Other existing tools typically use complex SPICE-based electrical characterization to model soft faults. For example, Rao et al. [11] model such faults by averaging glitch waveforms defined by Weibull probability distributions. Reliability evaluation becomes prohibitively

expensive if small changes require enumeration of all paths in a circuit, or re-computation of an entire decision diagram. BDD-based and symbolic techniques [5, 14, 7, 4] are unacceptably slow or require too much memory, as we show in Section 6. Some existing tools only work with a single process technology and very small gate libraries [14, 11].

AnSER uses a probabilistic logic-level fault model for both single and multiple faults to reason efficiently about the resulting errors. As clock frequency increases and threshold voltages decrease, logical masking also tends to dominate over electrical and timing masking. Hence reliability optimization need not be delayed until layout and electrical information are available. By leveraging fast bit-parallel simulation, AnSER offers linear-time SER evaluation and fast incremental updates after circuit transformations.

The fault model in question is based on the standard stuck-at (SA) fault model. For every clock cycle, we assume that each circuit node g has a *temporary single stuck-at-1* (TSA-1) fault with occurrence probability $Perr_1(g)$ if g is 0, and a temporary single stuck-at-0 (TSA-0) fault with $Perr_0(g)$ otherwise. While the TSA model focuses on logic masking, it can also incorporate the other masking mechanisms if desired. For example, electrical masking can be approximated by derating $Perr_0$ and $Perr_1$ by a factor dependent on adjacent gates [11]. Zhang et al. [14, 7] demonstrate the incorporation of timing masking by dividing error probabilities by a constant dependent on the clock period.

Using the TSA fault model, AnSER computes the SER of the entire circuit as a *probability of error* per cycle, by considering primarily logic masking. The results can easily be converted into units of FIT, or failures per 10^9 seconds. If the soft error probability per cycle is p, then the expected number of failures per 10^9 seconds is simply $p \times freq \times 10^9$ where $freq$ is the clock frequency. Assuming only one error occurs in each cycle, $Perr_0(g)$ is the probability that only gate g has an error. Therefore, gate SERs in units of FITs can be used in similar fashion.

We also consider *temporary multiple stuck-at* (TMSA) faults. Here, it is assumed that for each node g, the probabilities $Perr_0(g)$ and $Perr_1(g)$ are independent. Hence the probability of two gates g_1 and g_2 experiencing a TSA-0 fault simultaneously is $Perr_0(g_1)Perr_0(g_2)$.

4.2 SER and Testability

Next, we describe how AnSER uses signatures and ODC masks to derive several metrics that are necessary for our SER computation. These metrics are based on the *controllability, observability and testability* parameters commonly used in ATPG [3]. In particular, controllability refers to the ability to set the value of a node to a 0 or a 1.

Figure 3 summarizes the algorithm used by AnSER for SER computation. It involves two topological traversals of the target circuit: one to propagate signatures forward and another to propagate ODC masks backwards. The ratio of 0s and 1s in a node's signature is taken as a measure of its controllability, while the relative proportion of 1s in an ODC mask indicates observability. These two measures are combined to obtain a testability figure-of-merit for each node of interest, which is then multiplied by the probability of the associated TSA to obtain the SER for the node.

We define the 1-*controllability* of node g, denoted $con_{i1}(g)$, as the fraction of 1s in the signature $sig(g)$:

$$con_1(g) = \text{ones}\big(sig(g)\big)/K \tag{1}$$

```
compute_TSA_SER(Circuit C, int K)
{
 topological_sort(C);
 Perr(C) = 0;
 for(all nodes g ∈ C)
  sig(g) = compute_sig(g);
 reverse_topological_sort(C);
 for(all gates g ∈ C)
  ODCmask(g) = compute_odc_mask(g);
 for(all nodes g ∈ C)
  test0(g) = zeros(sig(g)&ODCmask(g))/K;
  test1(g) = ones(~ sig(g)&ODCmask(g))/K;
  Perr(C)+ = (Perr0(g)test1(g)+Perr1(g)test0(g));
 return Perr(C);
}
```

Figure 3: The SER computation algorithm for TSA faults.

The corresponding *0-controllability* metric is $con_0(g) = 1 - con_1(g)$. The *observability* of a node is defined as the number of 1s in its ODC mask.

$$obs(g) = \text{ones}\big(ODCmask(g)\big)/K \tag{2}$$

This observability metric is an estimate of the probability that g's value is propagated to a primary output. The 1-*testability* of g, denoted $test_1(g)$, is the number of bit positions where g's ODC mask and signature both are 1.

$$test_1(g) = \text{ones}\big(sig(g)\&ODCmask(g)\big)/K \tag{3}$$

Similarly, 0-*testability* is the number of positions where the ODC mask is 1 and the signature is 0. In other words, 0-testability is an estimate of the number of vectors that test for stuck-at-0 faults.

Example 2 *Consider again the circuit in Figure 2. Node g has signature $sig(g) = 01011011$ and ODC mask $ODCmask(g) = 01000100$. Hence, $con_1(g) = \text{ones}(sig(g)) = 5/8$, $con_0(g) = 3/8$, $obs(g) = 2/8$, $test_0(g) = 1/8$ and $test_1(g) = 1/8$.*

Suppose each node g in a circuit C has fault probabilities $Perr_0(g)$ and $Perr_1(g)$ for TSA-0 and TSA-1 faults, respectively. Then the SER of C is the sum of SER contributions at each gate g in the circuit. Here, we weigh gate error probabilities by the testability of the gate for the particular TSA.

$$Perr(C) = \sum_{g \in C} test_1(g)Perr_0(g) + test_0(g)Perr_1(g) \tag{4}$$

Example 3 *The $test_0$ and $test_1$ measures for all of the nodes in the circuit are given in Figure 2. If each gate has TSA-1 probability $Perr_0 = p$ and TSA-0 probability $Perr_1 = q$, then the SER is given by $Perr(C) = 2p + (13/8)q$.*

The metrics $test_0$ and $test_1$ implicitly incorporate error sensitization and propagation conditions, Hence Eq. 4 accounts for the possibility of an error being logically masked. Note that the $Perr_0(g)$ refers to the 1-controllability of g and so is weighted by the 1-testability; similarly for $Perr_1(g)$.

4.3 Node Impact Analysis

In order to identify areas of the circuit for resynthesis, we need to assess the impact of individual nodes on the circuit's SER. Intuitively, a nodes's influence on SER is proportional

```
compute_impact(Circuit C, node n)
{
  F = fanin_cone(n,C);
  impact(n) = 0;
  for(all gates g ∈ F)
    impact(n) += Perr0(g)reltest1(g,n);
    impact(n) += Perr1(g)reltest0(g,n);
  return impact(n);
}
```

Figure 4: The *impact* computation algorithm for TSA faults.

to the probability that faults arrive at the node, and the probability that those faults are observed as errors at the output. In other words, a node has high impact if many observable faults "flow" through it.

The proposed algorithm for computing impact is shown in Figure 4. It employs a notion of the testability of one node g relative to another node n, embodied in the definitions of $reltest_1(g,n)$ and $reltest_0(g,n)$.

$$reltest_1(g,n) = \text{ones}((ODCmask(g)\&ODCmask(n))\&sig(f))/K$$

In general, nodes closer to the primary outputs are more observable than those closer to the primary inputs. However, a node g in the fanin-cone C of node n may have observability greater than n due to fanout in C. Thus, we can mask the ODC of node g by the ODC mask of n in order to compute relative testability. The *impact* of n on the overall SER is then calculated as:

$$impact(n) = \sum_{g \in fanin(n)} Perr_0(g)reltest_1(g,n) + Perr_1(g)reltest_0(g,n)$$

Example 4 *For the circuit in Figure 2, $reltest_1(g,h)$ is given by* $\text{ones}(sig(g)\&(ODCmask(g)\&ODCmask(h)))/K$ *which is* $\text{ones}\big(01011011 \;\&\; (01000100 \;\&\; 01110111)\big)/8 = 1/8$. *Suppose that each gate has a TSA probabilities $Perr_1 = p$ and $Perr_0 = q$. Then, the impact of h is $q/8 + p/8 + q/8 + 4p/8 = q/4 + 5p/8$.*

If a subcircuit C' is hardened against TSA faults, the number of errors flowing through C' will decrease, as will the *obs* metric of nodes in C''s fanin cone. Therefore, once local design changes are made, the testability measures are incrementally updated by updating the corresponding signatures and ODC masks in C''s fanin and fanout cones.

4.4 Handling Multiple Errors

We now consider multiple simultaneous errors using the TMSA fault model. As pointed out in [4], error cancellation due to mutual masking can change error propagation conditions. AnSER addresses TMSA faults by propagating joint signal controllability, while considering cumulative error probabilities on groups of correlated signals, as explained in Figure 5. For a given set of gates S (not only pairs as in [4]), we compute the output error probability $Perrin_O(S)$ by considering the probability of each input combination (including cumulative input errors) $Perrin_I(S)$, and gate errors $Perr_E(S)$. The SER is derived by adding the error probability for each output combination thus:

$$Perr(C) = \sum_O |Perrin_O(C) - con_O(C)| \qquad (5)$$

In the case of fanout-free circuits, each set of gates s is just a single gate since no signal correlations exist. Then the SER calculation reduces to propagating cumulative error probabilities and error correlations forward through the circuit, gate by gate. This calculation runs in linear time. In circuits with reconvergent fanout, AnSER uses pairs of gates in order to capture pairwise signal correlations. Joint controllabilities are also computed using signatures, e.g., the probability of $x = 1, y = 1$ is

$$con_{11}(x,y) = \text{ones}\big(sig(x)\&sig(y)\big)/K \qquad (6)$$

The law of large numbers implies convergence to correct joint distributions as the number of simulation vectors grows.

```
compute_output_probs(gate sets S)
{
  topological_sort(S);
  for(all gates sets s ∈ S)
    for(all input combinations I)
      for(all gate error combinations E)
        O = erroneous_output(I,E);
        Perrin0(s) += Perrin_I(s) * Perr_E(s);
}
```

```
compute_TMSA_SER(Circuit C, int K)
{
  S = gates_with_common_sinks(C);
  compute_output_probs(S);
      for(all output combinations O)
        Perr(C) += |Perrin0(C) − con0(C)|;
    return Perr(C);
}
```

Figure 5: The SER computation algorithm for TMSA faults.

5 Synthesis for Reliability

We now present two methods for logic synthesis that leverage the fast reliability evaluation methods embodied in AnSER to improve the reliability of a given circuit.

5.1 Covering-based Resynthesis

The first method increases logic masking at high-impact nodes by exploiting redundancy already present in the circuit as identified by *covering relationships* among existing nodes. Compared to techniques such as *partial TMR* that replicate vulnerable signals, it incurs a smaller area overhead as it increases logic masking through the addition of single gates.

We say that g *covers* f, denoted $f \subseteq g$, if g is 1 for every input vector that makes $f = 1$ (here we are equating nodes with the Boolean functions they realize in the usual manner). In the presence of observability don't-cares, this relation can be generalized (using bitwise operations) to:

$$f\&C(g) \subseteq g\&C(g) \qquad (7)$$

Here $C(g) = \sim ODC(g)$, is the Boolean function representing the care set of g. In other words, g covers f if and only if g is 1 or a don't-care wherever f is 1. We define node g to be an *anti-cover* of node f when:

$$g\&C(g) \subseteq f\&C(g) \qquad (8)$$

The *impact* measure defined in Section 4.3 is used to select areas of the circuit for redesign. For a high-impact node

x, we find other nodes that it covers or anti-covers. Given a candidate node y covered by x , one can add redundant logic by transforming node x into $OR(x,y)$ because $y \subseteq x$ implies $OR(x,y) = x$. Similarly, if x is an anti-cover of y, we transform node x into $AND(x,y)$. To generalize, we identify y such that $x = OP(x,y)$.

In the trivial case where x is chosen as a candidate cover for itself, the redundant logic generated by $x = OP(x,x)$ will not lead to reliability improvements. At the other extreme, if x and y have disjoint fanin cones and $x = y$, then all errors that cause x to flip from 0 to 1 will be masked when x is replaced by $AND(x,y)$. Similarly, all 1-to-0 errors will be masked by $OR(x,y)$. In the general case of $x = OP(x,y)$ where x and y are different nodes, the impact of x and the portion of its fanin that is disjoint from y will be reduced as determined by OP. This occurs because sensitized paths in the fanin cone that include x but not y will benefit from the extra logic masking generated by $OP(x,y)$.

The covering relation can be extended naturally to signatures and bit-parallel simulation. For instance, suppose x has signature $sig(x) = 11000$ and $sig(y) = 11001$. By definition, $sig(x) \subseteq sig(y)$, therefore x can be replaced by $AND(x,y)$. In this case, all 0-to-1 flips of the third and fourth input vectors will be masked, as long as they are not propagated through both x and y. If y is replaced by $OR(x,y)$ then all 1-to-0 flips of the first two bits will be masked.

Since AnSER maintains signatures and ODC information for each node, we can quickly find covers for resynthesis. If a node has multiple covers, we break ties using *impact* of candidate nodes. Figure 6b illustrates replicated logic for node a derived by utilizing don't-care values stored with its signature. Signature-based replication must be verified since signatures do not fully capture Boolean functions. We use a SAT solver (MiniSAT) to check equivalence by constructing miters along a cut in the fanout cone of x between the original circuit and the new circuit with cover $OP(x,y)$; for further details, see [9].

5.2 Local Rewriting

Our second method for synthesis guides *logic rewriting* to optimize area and reliability simultaneously. Rewriting is a general technique that optimizes small subcircuits to obtain overall area improvements [6]. The implementation reported in [1, 6] first derives a 4-input cut for a selected node, defining a one-output subcircuit. Functionally-equivalent replacement candidates are then found using look-up tables.

To extend algorithms described in [1] to improve reliability *and* area, we rewrite 4-input subcircuits such that their reliability improves locally. To ensure global reliability improvement, we re-simulate the circuit and update SER estimates. Computational efficiency is ensured through fast incremental updates by AnSER. Figure 6a illustrates two candidate rewrites, of which the larger circuit exhibits greater logic masking.

The inherent trade-off between reliability and testability suggests that our synthesis techniques may complicate post-manufacturing test. However, our method maintains testability and signature information for each node with respect to given input vectors. Therefore, by avoiding mergers between nodes with identical signatures, one can ensure that every fault is testable by some of existing input vectors.

6 Empirical Validation

We report empirical results for reliability evaluation using AnSER and our two reliability-driven synthesis techniques.

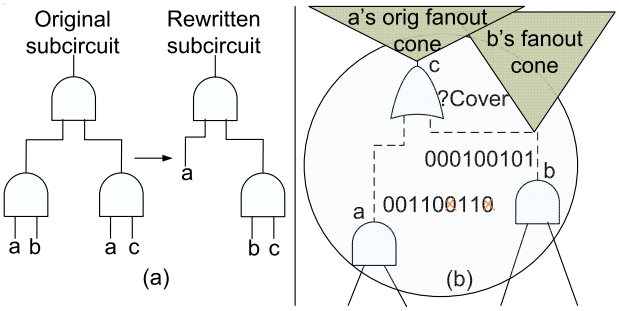

Figure 6: (a) Rewriting a subcircuit to improve area. (b) Finding a candidate cover for node a.

The experiments were conducted on a 2.4 GHz AMD Athlon 4000+ workstation with 2GB of RAM. The algorithms were implemented in C++.

In the spirit of traditional comparisons to Monte-Carlo, we first compared our AnSER algorithm under the TSA fault model with complete test-vector enumeration using the ATPG tool ATALANTA. We provided ATALANTA with a list of possible stuck-at faults in the circuit to generate tests in "diagnostic mode" which generates all test vectors for each fault. Since TSA faults are SA faults that last only for one cycle, the probability of a TSA fault causing an output error is equal to the number of test vectors for the corresponding SA fault weighted by their frequency. Assuming uniform input distribution, the fraction of vectors that detect a fault provides an exact measure of its testability. Then, we computed the SER by weighting the testability with a small gate error probability as in Eq. 4. While the exact computation can be performed only for small benchmarks, Table 1 suggests that our algorithm is accurate to about 3% for $2,048$ simulation vectors. More vectors can be used if desired.

To obtain accurate gate information for the experiments, we adapted data from [11], where several gate types are characterized in a 130nm, $1.2V_{DD}$ technology via SPICE. By using an average SER value of 4×10^{-7} for all gates, we are able to match the numerical results from [11]. While the reliability evaluators in [15, 14, 11, 4] report error rates that differ by orders of magnitude, AnSER can be calibrated to a given dataset. Table 2 compares AnSER with the prior art on ISCAS-85 benchmarks, using similar or identical CPUs. While the runtimes in [4] include 50 runs, the runtimes in [11] are reported per input vector. Thus we multiply data from [11] by the number of vectors $(2,048)$ used there. Our runtimes appear better by several orders of magnitude.

Table 3 shows improvements in SER and area overhead due to covering-based resynthesis. The first set of results are for exact covers, i.e., covers that do not consider ODCs,

Circuit	No. Gates	ATALANTA	AnSER	% Error
c17	13	6.96E-07	6.96E-07	0.01
majority	21	6.25E-06	6.63E-06	6.05
decod	25	2.60E-05	2.62E-05	0.83
b1	25	1.28E-05	1.31E-05	2.81
pm1	68	2.86E-05	3.00E-05	4.70
tcon	80	5.30E-05	5.39E-05	1.67
x2	86	3.78E-05	3.87E-05	2.20
z4ml	92	5.29E-05	5.37E-05	1.50
parity	111	7.60E-05	7.69E-05	1.24
pcle	115	5.38E-05	5.34E-05	0.75
pcler8	140	7.06E-05	7.24E-05	2.52
mux	188	1.58E-05	1.38E-05	12.54
Avg				3.06

Table 1: SER (FIT) data from AnSER and ATALANTA.

Circuit	Gates	Runtime (s)			
		AnSER	SERD[15]	FASER [14]	[4]
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: Runtime comparisons of reliability evaluators.

Circuit	SER	Area	With exact covers		With approx covers	
			% improv SER	% area ovrhead	% improv SER	% area ovrhead
cordic	5.334 E-5	84	1.7	1.2	27.3	45.2
b9	1.89E-5	114	18.1	14.9	30.7	31.6
C432	1.39E-3	215	37.6	14.0	38.7	14.9
C880	5.17E-5	341	9.6	0.9	13.1	2.3
C499	4.24E-4	432	1.0	3.2	32.2	20.6
C1908	1.92E-4	432	5.9	9.0	32.4	24.1
C1355	1.09E-2	536	25.3	9.0	30.7	8.6
alu4	6.12E-4	740	55.9	0.9	55.9	1.6
i9	1.66E-4	952	65.4	6.6	65.4	6.6
C3540	2.38E-3	1055	31.1	2.2	49.4	3.6
dalu	3.08E-4	1387	74.3	1.2	74.3	1.2
i10	1.0E-4	2824	40.4	5.4	40.4	5.6
des	9.84E-5	4252	11.4	2.9	26.7	4.4
Average			**29.1**	**5.5**	**39.8**	**13.1**

Table 3: Improved SER with covering-based resynthesis.

while the second uses ODCs to increase the number of candidates. In both cases AND/OR gates are used according to the covering relationship. For exact covers, we average 29.1% SER improvement with only 5% area overhead. The improvements for the ODC covers are 39.8% with area overhead of 13.1%, suggesting a greater gain per additional unit area than in *partial TMR* techniques [8] which achieve a 91% improvement but increase area by $1,04\%$.

Table 4 illustrates the use of AnSER to guide the local rewriting implementation in the ABC logic synthesis package [1]. AnSER calculates the global reliability impact of each local change to decide whether to accept this change. After checking hundreds of circuit rewriting possibilities, those that improve SER and have limited area overhead are retained. The data indicate that, on average, SER decreases by $10,7\%$, while area decreases by 2.3%. For instance, for `alu4`, a circuit with 740 gates, we achieve 29% lower SER, while reducing area by 0.5%. Although area optimization is often thought to hurt reliability, results show that carefully guided logic transformations can eliminate this problem.

Circuits	SER	Area	No. rewrites	%improv SER	%area decrease	Time (s)
alu4	6.12E-4	740	13	29.3	0.5	24.5
b1	8.62E-6	14	0	0.0	0.0	0.2
b9	1.89E-5	114	8	6.8	0.9	0.3
C1355	1.09E-2	536	97	1.2	9.0	37.6
C3540	2.38E-3	1055	23	5.8	0.9	51.5
C432	1.38E-3	215	68	5.5	1.4	12.1
C499	4.23E-4	432	37	0.0	0.5	13.0
C880	5.17E-5	341	7	0.2	0.0	5.4
cordic	5.33E-5	84	5	1.2	1.2	0.5
dalu	3.08E-4	1387	58	24.0	3.2	35.0
des	9.84E-5	4252	282	11.2	0.1	12.3
frg2	1.98E-5	1228	96	27.9	2.0	8.9
i10	2.00E-4	2824	143	5.0	0.6	16.7
i9	1.66E-4	952	83	31.4	11.7	35.3
Average				**10.7**	**2.3**	**18.1**

Table 4: Improvements in SER and area with local rewriting.

7 Conclusions

We have presented a technology-independent reliability evaluator AnSER designed for use in logic synthesis. AnSER achieves very high speed by efficient use of node signatures and ODC masks. We also described a method for identifying high-impact areas of a circuit to target for restructuring. Fault models with single and multiple errors per cycle are supported. Empirically, our SER evaluator runs 2-to-3 orders of magnitude faster than the prior art. We also proposed a new, reliability-aware resynthesis strategy that replicates a vulnerable node by adding a single gate. This strategy manipulates previously-stored signatures to find ODCs. On average, it improves reliability by some 29-40% with only 5-13% area overhead. Finally, we successfully applied AnSER to local rewriting, and showed that this approach simultaneously improves area and reliability.

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

References

[1] Berkeley Logic Synthesis and Verification Group, "ABC: A System For Sequential Synthesis & Verification", `http://www.eecs.berkeley.edu/~alanmi/abc/`

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

[3] M. Bushnell, V. Agrawal, *Essentials of Electronic Testing*, Kluwer, 2000, pp. 129-150.

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

[5] S. Krishnaswamy, G. F. Viamontes, I. L. Markov, J. P. Hayes, "Accurate Reliability Evaluation and Enhancement via Probabilistic Transfer Matrices", *Proc. DATE 2005*, pp. 282-287.

[6] A. Mishchenko, S. Chatterjee, R. Brayton, "DAG-aware AIG rewriting: A Fresh Look at Combinational Logic Synthesis", *Proc. DAC 2006*, pp. 532-535.

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

[8] K. Mohanram, N. A. Touba, "Partial error masking to reduce soft error failure rate in logic circuits" *Proc. DFT 2003*, pp. 433-440.

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

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

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

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

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

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

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

[16] Q. Zhu, N. Kitchen, A. Kuehlmann, A. Sangiovanni-Vincentelli, "SAT sweeping with Local Observability Don't-cares", *Proc. DAC 2006*, pp. 229-234.