Signature-based SER Analysis and Design of Logic Circuits

Smita Krishnaswamy [†], Stephen M. Plaza [♯], Igor L. Markov [*], and John P. Hayes [*]
[†] IBM T.J. Watson Research Center, 1101 Kitchawan Rd., RT 134, Yorktown Heights, NY 10598
[♯] Advanced Technology Group, Synopsis Inc., 2025 NW Cornelius Pass Rd, Hillsboro, OR 97124
[*] University of Michigan, EECS Department, Ann Arbor, MI 41809
{smita,splaza,imarkov,jhayes}@eecs.umich.edu

***Abstract*—We explore the use of signatures, i.e., partial truth tables generated via bit-parallel functional simulation, during soft error analysis and logic synthesis. We first present a signature-based CAD framework that incorporates tools for the logic-level *A*nalysis of *S*oft *E*rror *R*ate (AnSER) and for *S*ignature-based *De*sign for *R*eliability (SiDeR). We observe that the SER of a logic circuit is closely related to various testability parameters, such as signal observability and probability. We show that these parameters can be computed very efficiently (in linear time) by means of signatures. Consequently, AnSER evaluates logic masking two to three orders of magnitude faster than other SER evaluators while maintaining accuracy. AnSER can also compute SER efficiently in sequential circuits by approximating steady-state probabilities and sequential signal observabilities. In the second part of the paper, we incorporate AnSER into logic synthesis design flows aimed at reliable circuit design. SiDeR identifies and exploits redundancy already present in a circuit via signature comparison to decrease SER. We show that SiDeR reduces SER by 40% with only 13% area overhead. We also describe a second signature-based synthesis strategy that employs local rewriting to simultaneously improve area and decrease SER. This technique yields 13% reduction in SER with a 2% area decrease. We show that combining the two synthesis approaches can result in further area-reliability improvements.**

I. Introduction

Soft (transient) errors are becoming an important concern in digital integrated circuits. It has long been known that many soft faults are masked and do not lead to observable circuit errors. Therefore, analyzers are needed to assess the impact of masking mechanisms on the soft error rate (SER) of a circuit. Further, deliberately increasing masking is key to low-SER designs. Hence, SER analysis can effectively guide and evaluate synthesis by accounting for relevant masking mechanisms.

In this paper, we present a methodology to guide the logic synthesis process towards greater design robustness. First, we develop an SER analyzer, *AnSER*, which estimates logic masking efficiently and accurately. When a fault occurs in a portion of the circuit that is logically un-sensitized, it is said to be "logically masked". This type of masking originates during logic synthesis and remains in effect through the rest of the design flow. The difficulty in estimating logic masking is the input (state) space explosion problem when considering the different paths of sensitization invoked by different input patterns (states). We use signature-based analysis to efficiently solve this problem.

A signature is a partial truth-table of a Boolean function, computed by bit-parallel functional simulation of the circuit on applying random inputs. Signatures allow us to compute probabilistic information about a circuit that is relevant to SER computation. First, we compute signatures for all nodes in a circuit. Then, we compute observability don't-care (ODC) masks from the signatures to estimate node observability and testability. This information is used, in turn, to compute the SER. Figure 1 outlines our SER-aware synthesis methodology, which exploits the intimate relations between logic masking, simulation signatures, ODCs, and the testability of stuck-at faults. We also extend our techniques to evaluate the impact of soft faults on sequential circuits. We find that signatures offer a way to overcome computational challenges associated with sequential-circuit analysis including steady-state probability computation, reachability analysis, and Markov-chain analysis, which often tax other analysis methodologies.

The second part of the paper focuses on design for decreased SER. Soft-error reliability can be improved by increasing masking opportunities. In the past, researchers have resorted to massive functional redundancy in schemes such as in triple modular redundancy (TMR) to improve reliability. However, these methods require a substantial increase in area and power consumption. In contrast, as we show, closely coupling synthesis and SER analysis can reduce overhead with significant improvements in reliability. We present a novel synthesis technique called *SiDeR* that uses signatures and ODCs to identify partial redundancies among critical nodes of the circuit. Then, SiDeR can protect logic in the fan-in cone of the critical node with the addition of a single gate. The paper's main contributions are as follows:

- A fast incremental SER analyzer, AnSER, that can be used stand-alone or integrated with logic synthesis
- A novel synthesis technique, SiDeR, that decreases SER by exploiting logical covering relationships and observability don't-cares

In addition, we demonstrate the use of AnSER in a general logic synthesis flow to guide a local restructuring technique known as *rewriting*. This technique rewrites small windows of logic throughout the circuit in order to improve area. The use of *AnSER* can simultaneously improve area *and* SER.

The paper is organized as follows. Section II discusses previous work on SER analysis and SER-aware synthesis. Section III covers background on bit-parallel simulation and signatures. Section IV introduces our SER analysis methodology and Section V extends the analysis to sequential circuits. Section VI describes two strategies for synthesis to decrease

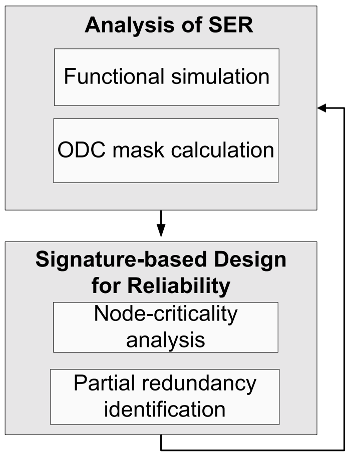
Analysis of SER
Functional simulation
ODC mask calculation
Signature-based Design for Reliability
Node-criticality analysis
Partial redundancy identification

Fig. 1. The proposed signature-based analysis framework for SER evaluation and synthesis.

SER. Section VII presents empirical results. Finally, Section VIII concludes the paper.

II. Previous Work

Recent SER evaluators include SERA [28], FASER [27], SERD [23], and MARS-C [18] along with its sequential extension MARS-S [19]. These tools estimate the SER of a technology-mapped circuit by accounting for three masking mechanisms with varying levels of detail. The three masking mechanisms are [24]: 1) logic masking (the glitch occurs in a non-sensitized portion of the circuit), 2) electrical masking (the glitch is attenuated and blocked by the electrical characteristics of CMOS gates), and 3) temporal masking (the glitch occurs in a non-latching portion of the clock cycle). Logic masking is accounted for by explicit enumeration of the input vector (or state) space in decision diagram-based methods [18], [27] or by fault simulation on specific vectors [23], [28]. Electrical masking is assessed using SPICE-based pre-characterization of the gate library. Timing masking is either approximated as a derating factor proportional to the latching time of flip-flops in the design [27], or based on timing analysis information [18]. In addition, MARS-S [19] uses Markov chain analysis and symbolic simulation to analyze SER in sequential circuits.

While these methods offer detailed analysis of SER, they can be difficult to use during logic design because: 1) they require complete information such as electrical characterization and timing analysis, which may be unavailable during logic design, and 2) they use unscalable methods for logic masking analysis. Some tools [13], [18], [27] use ADDs (DDs with multiple real valued-terminals) to completely enumerate input patterns and calculate pattern-dependent error probabilities for logic masking analysis —this has exponential worst-case complexity. This use of ADDs in SER analysis is different from the use of BDDs in logic synthesis to represent Boolean functions. The latter is generally much more efficient. Other tools electrically simulate circuits vector-by-vector, which can slow down SER analysis and become a bottleneck in circuit optimization as well.

Several techniques are known to reduce the impact of soft errors on logic circuits. Rao et al. [22] use the algorithm from [23] to selectively resize gates and flip-flops. Low-energy particle strikes are less likely to cause a glitch in larger gates due to the increased internal capacitance of the gate. Larger gates also imply that glitches are less likely to appear at gate outputs, and those that do appear are often electrically masked.

Soft errors can also be mitigated by adding redundant logic. Classic techniques such as TMR and quadded logic [25] achieve this by systematically replicating logic. In quadded logic, each gate is replaced by a network of four gates which logically mask single faults. TMR triplicates the entire circuit and uses voters to mask faults. Mohanram and Touba [20] reduce the cost of TMR by replicating only the most susceptible gates. However, even partial replication of this kind is quite expensive.

Almukhaizim et al. [2] proposed SER reduction via guided rewiring. In the form of rewiring that they use [26], one of four design errors is introduced to the circuit at each step. These include: 1) removing a target wire, 2) changing the gate driven by the target wire, 2) adding an extra input to the gate driven by the target wire, and 3) replacing the target wire with a different wire. Then, the algorithm from [26] is called to provide a list of possible single-operation corrections for the introduced errors. The correction that improves SER by the most is chosen based on re-evaluation of SER by the tool SERA [28]. ATPG-based rewiring [5] has been used in other contexts such as in optimizing sequential circuits [16], and timing optimization [8]. However, the work in [2] appears to be the first example of reliability-guided circuit restructuring without the explicit addition of redundancy.

Almukhaizim and Makris have recently [3] proposed improving SER through the addition of redundant wires. Redundant wires are identified by deriving relations between wires using logical implication. Others have used logic implication to add (and remove) redundancies in circuits for logic optimization [14], [16]. The approach of [3] related to our SiDeR approach, originally proposed in [12]. However, in contrast to logic implication analysis, we identify redundancy using signature comparison. Empirical results show that our methods improve SER by a wider margin.

III. Signatures and ODC Masks

In this paper, we systematically use node signatures to compute the SER, to target error-sensitive areas of a circuit, and to identify redundant nodes for resynthesis. A circuit node g can be labeled by a *signature* $sig(g) = F_g(X_1)F_g(X_2)\ldots F_g(X_K)$ 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 (exhaustively simulating) generates a signature that corresponds to a full truth table. In general, $sig(g)$ can be seen as a kind of ”supersignal” appearing on g. It is composed of individual binary signals that are defined by some current set of vectors. Like the individual signals, $sig(g)$ can be processed by EDA tools such as simulators and synthesizers, e.g., it is a single entity. It can be propagated through a sequence of logic gates and combined with other signatures via Boolean operations. This processing can take advantage of bitwise operations to speed up the overall computation compared to processing the signals that compose $sig(g)$ one at a time. Signatures with thousands of bits can be useful in pruning non-equivalent nodes during equivalence checking [21], [29]. A related speedup technique is also the basis for "parallel" fault simulation [4]. The basic algorithm for computing signatures is shown for reference in Figure 2. Here, $Op < g >$ refers to the operation gate g. This operation is applied to the signatures of the input nodes of gate g, denoted $inputsigs(g)$.

```
compute_sigs(Circuit $\mathbf{C}$, size $\mathbf{K}$)
{
  for(all inputs $\mathbf{i} \in \mathbf{C}$)
    $\mathbf{sig(i) = gen\_random\_sig(K)}$
  $\mathbf{sort\_topological(C)}$
  for(all nodes $\mathbf{g} \in \mathbf{C}$)
    $\mathbf{sig(g) = Op < g > (inputsigs(g))}$
}
```

Fig. 2. An algorithm for signature computation.

Figure 3 shows a 5-input circuit where each of the 10 nodes is labeled by an 8-bit signature SIG computed with eight input vectors. These vectors are randomly generated, and conventional functional simulation propagates signatures to the internal and output nodes. In a typical implementation such as ours, signatures are stored as logical words and manipulated with 64-bit logical operations, ensuring high simulation throughput. Therefore 64 vector simulations are conducted in parallel with each signature processed. 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, $ODCmask(g) = X_1 \notin ODC(F_g) X_2 \notin ODC(F_g) \ldots X_K \notin ODC(F_g)$. The ODCmask is computed by bitwise negating $sig(g)$ and re-simulating the circuit through the fan-out of g to check if the changes are propagated to any of the primary outputs. This algorithm is shown as $compute\_odc\_exact$ in Figure 4 and has complexity $O(N^2)$ for a circuit with N gates. Its practical implementations may truncate re-simulation at gates where signatures do not change, thus achieving an additional speed-up.

We found the heuristic algorithm from [21], which has only $O(N)$ complexity to be particularly convenient to use. This algorithm is also shown in Figure 4. Here, the circuit traversed in reverse topological order and, for each node, a local ODC mask is computed for its immediate downstream gates. The local ODC mask is derived by flipping each value in the input signatures of a gate to see if the output of the gate changes. The local ODC mask is then bitwise-ANDed with the respective global ODC mask at the output of the gate to produce the ODC

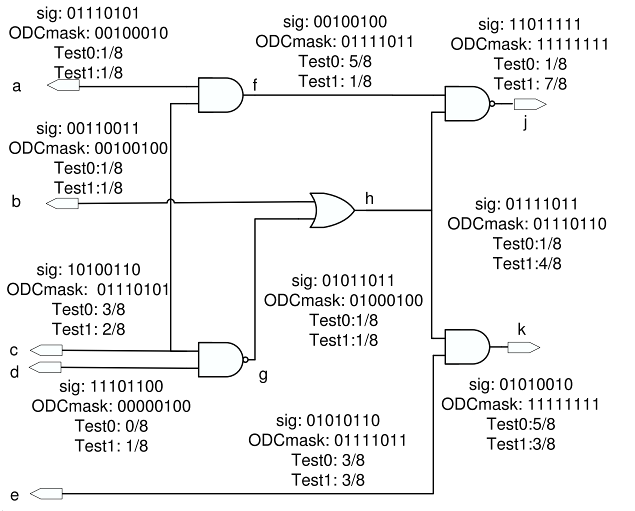

Fig. 3. Signatures, ODC masks, and testability information associated with circuit nodes.

mask of the gate *for* a particular fan-out branch. The ODC masks for all fan-out 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 fan-out branches. Reconvergent fan-out can eventually lead to incorrect values. The masks can be corrected by performing exact simulation downstream from the converging nodes. This step is not strictly necessary for SER evaluation as we show later.

Example 1: Figure 3 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, 01110001, of c on paths through f. Similarly, the local ODC mask of c for the gate with output g is 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.

IV. Analysis of SER

We now present the SER analyzer AnSER, which was specifically designed for use in logic synthesis. In this section, we focus on combinational logic, in the next section we cover SER in sequential circuits.

A. Fault Model for Soft Errors

Integrating SER analysis efficiently into logic synthesis requires scalability and logical-level 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. [23] model such faults by averaging glitch waveforms defined by Weibull probability distributions. Some existing tools only work with a single process technology and very small gate libraries [23], [27].

```
compute_odc_exact(Circuit $\mathbf{C}$, size $\mathbf{K}$)
{
 $\mathbf{compute\_sigs(C,K)}$
 $\mathbf{sort\_reverse\_topological(C)}$
 for(all nodes $\mathbf{g \in C}$)
  $\mathbf{newsig(g) = sig(g)}$
  $\mathbf{recompute\_sigs(C,K,g)}$
  for(each output $\mathbf{o \in C}$)
   $\mathbf{odc(g)| = newsig(o) \oplus sig(o)}$
  $\mathbf{restore\_computed\_sigs(C)}$

}
```

(a)

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

(b)

Fig. 4. (a) Exact and (b) approximate ODC mask computation algorithms.

```
compute_TSA_SER(Circuit $C$, int $K$)
{
 $\mathbf{sort\_topological(C)}$
 $\mathbf{compute\_sigs(C,K)}$
 $\mathbf{compute\_odc\_approx(C,K)}$
 for(all nodes $\mathbf{g \in C}$)
  $\mathbf{test_0(g) = zeros(sig(g)\&ODCmask(g))/K}$
  $\mathbf{test_1(g) = ones(\sim sig(g)\&ODCmask(g))/K}$
  $\mathbf{Perr(C){+}= Perr0(g)test_1(g)}$
  $\mathbf{Perr(C){+}= Perr1(g)test_0(g)}$
 return $\mathbf{Perr(C)}$
}
```

Fig. 5. The SER computation algorithm for TSA faults.

AnSER uses a probabilistic logic-level fault model for single 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 SER optimization need not be delayed until layout and electrical information are available. By leveraging fast bit-parallel simulation, AnSER offers linear-time SER analysis and fast incremental updates after circuit transformations.

We propose a fault model 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 $Perr_1(g)$, and a temporary single stuck-at-0 (TSA-0) fault with probability $Perr_0(g)$ otherwise. While this 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 [23]. Zhang et al. [18], [27] 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 FIT is simply $p \times freq \times 10^9$ where $freq$ is the clock frequency. Assuming only one fault occurs in each cycle, $Gerr_0(g)$ is the FIT rate of the gate g, and is related the probability of error $Perr_0(g)$, by a constant.

B. SER Evaluation

AnSER computes the SER by counting the number of test vectors that propagate the effects of a soft fault to the output(s). Test-vector counting was also used in [10] to compute SER, although the authors also used BDD-based techniques. Intuitively, if a large number of test vectors are applied at the inputs, then faults will be propagated to the outputs often. It should be noted that SER computation is inherently more difficult than test generation. Testing involves generating a vector that sensitizes the fault on a node, and propagates the resulting error to the output. SER evaluation involves counting the number of vectors that detect each fault and is therefore in the $\sharp P$-hard complexity class.

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 signal probability (controllability), observability and testability parameters commonly used in ATPG [4].

Figure 5 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 are taken as a measure of signal probability, 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. This SER for the node captures the probability that a fault occurs and its effects are propagated to the output. Our estimate can be contrasted with technology-dependent SER estimates that include timing and electrical masking.

We define the probability of node g having logic value 1, denoted $P[g = 1]$, as the fraction of 1s in the signature $sig(g)$:

$$P[g = 1] = ones\big(sig(g)\big)/K \qquad (1)$$

The corresponding *0-controllability* metric is $P[g = 0] = 1 - P[g = 1]$. The *observability* of a node is defined as the number of 1s in its ODC mask.

$$P[obs(g)] = ones\big(ODCmask(g)\big)/K \qquad (2)$$

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 $P[test_1(g)] = P[obs(g), g = 1]$, is the number of bit positions where g's ODC mask and signature both are 1.

$$P[test_1(g)] = \text{ones}\big(sig(g)\&ODCmask(g)\big)/K \qquad (3)$$

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 3. Node g has signature $sig(g) = 01011011$ and ODC mask $ODCmask(g) = 01000100$. Hence, $P[g = 1] = \text{ones}(sig(g)) = 5/8$, $P[g = 0] = 3/8$, $P[obs(g)] = 2/8$, $P[test_0(g)] = 1/8$ and $P[test_1(g)] = 1/8$.

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 weight intrinsic gate fault probabilities by the testability of the gate for the particular TSA.

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

Example 3: The $test_0$ and $test_1$ measures for each gate in the circuit are given in Figure 3. 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 fault sensitization and propagation conditions, Hence Equation 4 accounts for the possibility of a fault 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 $Perr_1(g)$.

V. SER Analysis in Sequential Logic

In this section, we extend our SER analysis to include sequential circuits, which have memory elements (D flop-flops) in addition to primary inputs and outputs. Recall that the values stored in flip-flops collectively form the state of the circuit. The combinational logic computes state information and primary outputs as a function of the current state and primary inputs. Below, we list three factors to consider while analyzing sequential circuit SER:

1) Steady-state probability distribution: It has been shown that under normal operation most sequential circuits exhibit convergence to particular state distributions [9]. Discovering the steady-state probability is useful for accurately computing the SER.
2) State reachability: Some states cannot be reached from a given initial state, therefore only the reachable part of the state space should account for the SER.
3) Sequential observability: Errors in sequential circuits can persist past a single cycle if captured by a flip-flop. A single fault may be captured by multiple flip-flops and result in multiple faults in subsequent cycles. Such faults can then be masked by logic.

The following two subsections develop a simulation-based framework to address these issues. In Section V-A, we perform steady-state and reachability analysis through sequential simulation. In Section V-B, we assess sequential observability by applying techniques from Section IV-B to time-frame-expanded circuits. In addition, we explain how these relatively simple solutions handle subtle concerns in sequential circuit SER analysis.

A. Steady-State and Reachability Analysis

Usually, the primary input distribution is assumed to be uniform, or is explicitly given by the user, while the state distribution has to be derived. Hachtel et al. [6], [9] show that aperiodic finite-state machines (FSMs) with strongly-connected state-spaces eventually reach a steady-state distribution. An FSM is *periodic* if its states can be visited only at regular intervals, and *aperiodic* otherwise. Periodic FSMs do not reach steady-state. A modulo-d counter is an example of such an FSM. In [6], it is shown that most ISCAS and other benchmark circuits reach steady state because they are *synchronizable*, in other words they can be taken to a reset state starting from any state, using a specific fixed-length sequence. This indicates that the circuits are aperiodic (otherwise different length sequences would have to be used from each state) and strongly connected (otherwise some states could not be taken to the reset state).

In order to approximate the steady-state distribution, we perform sequential simulation using signatures. Assume that a circuit with m flip-flops $L = \{l_1, l_2 \ldots l_m\}$ is in state $S_L = \{s_0, s_1, s_2 \ldots s_m\}$ where each $s_i \in \{0, 1\}$. Our method starts in state S_0 for each simulation (sets of 64 are conducted in parallel). Then, we simulate the circuit for n cycles. Each cycle propagates signatures through the combinational logic and stops when flip-flops are reached. Primary input values are generated randomly from a given distribution. At the end of each simulation cycle, flip-flop inputs are transferred to flip-flop outputs, which are in turn fed into combinational logic for the subsequent cycle. All other intermediate signatures are erased before the next simulation cycle starts. The K-bit signatures of flip-flops at the end of n simulations cycles together define K states. We claim that for a large enough n, these states are sampled from a steady-state probability distribution. Empirical results suggest that most ISCAS benchmarks reach steady-state in 10 cycles or less [19] under the above operating conditions.

Our method can also handle systems that are *decomposable*. Such systems pass through some transient states and are then confined to a set of strongly connected closed (SCC) states. That is, states in the system are partitioned into transient states and sets of SCC states. For such systems, the steady state is heavily dependent on the initial states. We address this implicitly performing by reachability analysis starting in a reset state. Thus, each bit of the signature corresponds to a simulation that (1) starts from a reset state and propagates through the combinational logic, (2) moves to adjacent reachable states, and (3) for large enough n, reaches steady-state within the partition.

Using our method, simulating a circuit with g gates for n simulation cycles, and K bit signatures takes time $O(Kng)$. Figure 6 summarizes our simulation algorithm for sequential circuits. Note that it does not require matrix analysis which is often the bottleneck in other methods. Markov matrices usually encode state transition probabilities explicitly, and can

```
seq_simulate(Circuit $C$, int $K$)
{
 for(all flip-flops $\mathbf{l} \in \mathbf{C}$)
  $\mathbf{output\_sig(l) = input\_sig(l)}$;
 for(all inputs $\mathbf{in_0} \in \mathbf{C}$)
  $\mathbf{in_0 = new\_random\_input()}$;
 $\mathbf{compute\_sigs(C, K)}$
}
```

Fig. 6. The algorithm for multi-cycle sequential circuit simulation.

be prohibitively expensive due to the problem of state space explosion [9], [19].

Figure 7 illustrates sequential simulation with 3-bit signatures. The flip-flops with outputs x and y are initialized to 000 in cycle 0, labeled T_0. Then the combinational logic is simulated. In T_1, the input of x and y are transferred to the output and the process continues. At the conclusion of the simulation, the values for x and y at T_3 are saved for sequential error analysis which is explained in the next subsection.

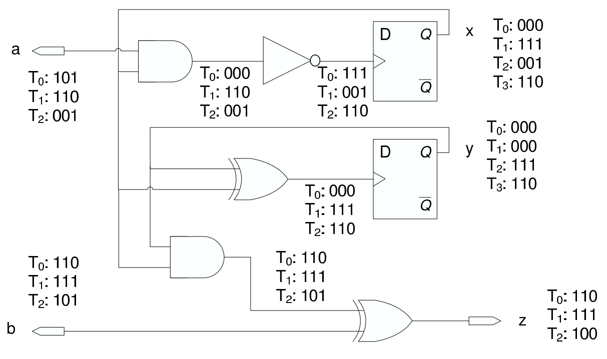

Fig. 7. Illustration of bit-parallel sequential simulation.

Although we only considered aperiodic systems, we observe that for a periodic system the SER would need to be analyzed for the maximum period D, since the state distribution oscillates over the period. If the FSM is periodic with period D, we can average over the SER for over D simulation cycles.

B. Error Persistence and Sequential Observability

In order to assess the impact of soft faults on sequential circuits, we consider several cycles through which faults persist using time-frame expansion. This involves making n copies of the circuit, $C_0, C_1 \dots C_{n-1}$, thereby converting a sequential circuit into a pseudo-combinational circuit. In the expanded circuit, flip-flops are treated as buffers. The outputs for the flip-flops of the k-th frame are connected to the primary inputs of frame $k+1$ frame (as appropriate) for $0 < k < n-1$. Flip-flop outputs that feed into the 0-th frame are treated as primary inputs and flip-flop inputs of frame n are treated as primary outputs. Figure 8 shows a three-time-frame circuit that corresponds to that of Figure 7. Here, the primary inputs and outputs of each frame are marked by their frame numbers. Further, new primary inputs and outputs are created corresponding to the inputs from flip-flops for frame 0 and outputs of flip-flops for frame 3. Intermediate flip-flops are represented by buffers (shaded).

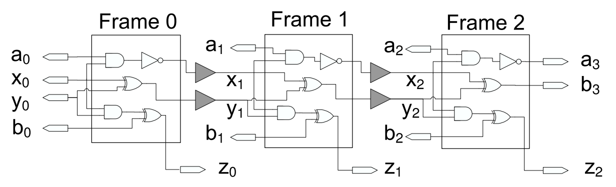

Fig. 8. Illustration of time-frame expansion into three frames C_0, C_1, C_2.

```
compute_seq_SER(Circuit $C$,int $K$,int $n$,int $f$)
{
 for($i < n$)
  $\mathbf{seq\_simulate(C, K)}$;
 $\mathbf{C' = time\_frame\_expand(C, f)}$
 $\mathbf{copy\_flipflop\_inputs(C', C)}$
 $\mathbf{compute\_sigs(C', K)}$
 $\mathbf{compute\_odc\_approx(C', K)}$
 for(all nodes $\mathbf{g} \in \mathbf{C_0}$)
  $\mathbf{test_0(g) = zeros(sig(g) \& ODCmask(g))/K}$
  $\mathbf{test_1(g) = ones(\sim sig(g) \& ODCmask(g))/K}$
  $\mathbf{Perr(C') += (Perr0(g) test_1(g) + Perr1(g) test_0(g))}$
 return $\mathbf{Perr(C')}$
}
```

Fig. 9. The SER computation algorithm for TSA faults in sequential circuits.

Observability is analyzed by considering all n frames together as a single combinational circuit, thus allowing the single-fault SER analysis described in the previous section to be applied to sequential circuits. Other useful information such as the average number of cycles during which faults persist can also be determined using time-frame expansion.

After the multi-cycle sequential simulation described in the previous section, we store the signatures of the flip-flops, and use signatures to stimulate the newly created primary inputs (corresponding to frame 0 flip-flops) in the time-frame expanded circuit. For instance, the x_0 and y_0 inputs of the circuit in Figure 8 are simulated with the corresponding signatures marked T_3 (the final signature after multi-cycle simulation is finished) from Figure 7. Randomly generated signatures are used for primary inputs not corresponding to flip-flops (such as a_0 and b_0 in Figure 8).

After simulation, we perform ODC analysis starting from the primary outputs and flip-flops inputs of the n-th frame all the way to the inputs of the 0-th frame. In other words, primary outputs and any flip-flops with errors after n cycles are considered to be observable. Figure 9 gives our algorithm for sequential SER computation. The value of n can be varied until the SER stabilizes, i.e., does not change appreciably from an n-frame analysis to an $(n+1)$-frame analysis.

The n-frame ODC-analysis can lead to different gates being seen as critical for SER. For instance, the designer can deem errors that persist longer than n cycles as more critical than errors that are quickly flushed at primary outputs. In this case, the ODC analysis only considers the fan-in cones of the primary outputs of C_n. The SER of the circuit with respect to n cycles of sequential masking is the SER computed on the C_0 frame as follows:

$$Perr(C_0) = \sum_{g_i \in C_0} P[test_1(g_i)]Perr_0(g_i) + P[test_0(g_i)]Perr_1(g_i) \tag{5}$$

The SER algorithm in Figure 9 still runs in linear time with respect to the size of the circuit, since each simulation is linear and ODC analysis (even with an n-frame analysis) runs in linear time as well. The SER values and run times of some ISCAS-89 benchmark circuits are given later (Section VII).

VI. Design for Reliability

We now present methods for logic synthesis that leverage the fast signature-based SER analysis method embodied in AnSER to improve the resilience of a given circuit with respect to soft errors. First, we discuss an SER-aware design method, *SiDeR*, which involves utilizing redundancy within the circuit identified using pre-computed signatures. This is a global restructuring technique in that connections can be made between nodes in any part of the circuit based on their informational redundancy. The second method locally restructures small portions of the circuit to improve area and SER. As we show in Section VII, these techniques can be combined or used individually.

A. Signature-based Design for Reliability

Our signature-based design for reliability (SiDeR) method is aimed at increasing logic masking at high-impact nodes by exploiting redundancy already present in the circuit. This redundancy is identified using signatures. Pairs of signatures are checked for functional relations, which when verified can be used to increase reliability through the use of a single gate. Compared to techniques such as partial TMR that replicate vulnerable signals, SiDeR incurs a smaller area overhead since it increases logic masking by adding gates one at a time.

In order to limit area overhead the functional relations that we consider only covering relationships between nodes. 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 the following.

$$f \& ODCmask(g) \subseteq g \& ODCmask(g) \tag{6}$$

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

$$g \& ODCmask(g) \subseteq f \& ODCmask(g) \tag{7}$$

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 $\text{AND}(x, y)$. 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 $\text{OR}(x, y)$, then all 1-to-0 flips of the first two bits will be masked.

For a node x, we find other nodes that it covers or anti-covers. Given a candidate node y covered by x, we add redundant logic by transforming node x into $\text{OR}(x, y)$ because $y \subseteq x$ implies $\text{OR}(x, y) = x$. Similarly, if x is an anti-cover of y, we transform node x into $\text{AND}(x, y)$. To generalize, we identify y such that $x = f(x, y_1, y_2 \ldots y_n)$ and f denotes an arbitrary Boolean function. Replacing x by f results in errors being masked for cases where x does not have a controlling value for f.

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

Signatures provide an especially effective method for identifying partial redundancy in the form of covering relationships. For instance, if $\text{OR}(x, y) = x$ then it follows that $sig(x) > sig(y)$ lexicographically (otherwise $sig(y)$ has a 1 in a position where $sig(x)$ does not). Therefore, sorting all of the signatures can narrow the search for candidate signals y. Also, $sig(x)$ must have more ones than $sig(y)$ so $|sig(x)| > |sig(y)|$ where $|sig(x)|$ is the *size* of the signature. This means that we can narrow the candidate set further by having a size-sorted list of signatures and intersecting the candidates found using the two lists. The search candidates can be pruned even more by performing multiple lexicographical sorts and multiple size sorts of signatures *starting from different bit positions*. For instance, if we sort the signatures lexicographically from the ith bit, $sig(x)$ must still occur before $sig(y)$ for the same reason. As a result, signature-based redundancy identification can be an efficient alternative to logic implication analysis which is used in [3].

1) Node Impact Analysis: In order to quickly identify candidates for resynthesis, we calculate a measure called *impact* that describes the node's influence on the overall circuit SER. Intuitively, this influence should be proportional 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 errors "flow" through it.

We propose a linear-time algorithm for computing *impact*, as shown in Figure 10. It employs a notion of the observability of one node g relative to another node f, embodied in the following definition of

$$relODCmask(g, f) = ODCmask(g) \& ODCmask(f) \tag{8}$$

As computed in Figure 10, the *impact* measure is precise in cases where all gate error probabilities are equal. The algorithm works by keeping a running signature called $impactsig(f)$ at each node f, which is an indication of the faults propagated to f through paths from its fan-out cone. In general, nodes closer to the primary outputs are more observable than those closer to the primary inputs. However, a node g in the fan-in cone F of node f may have observability greater than f due to fan-out in F. For the circuit in Figure 3,

```
compute_impact(Circuit C)
{
 sort_topological(f, C)
 for(all gates $f \in C$)
  for(all fanout stems $g \in inputs(f)$)
   $\mathbf{impactsig(f)| = impactsig(g)\&odc(f)}$
  $\mathbf{impactsig(f)| = relODCmask(g, f)}$
  $\mathbf{impact(f) = Perr * ones(impactsig(f))/K}$
}
```

Fig. 10. The algorithm for computing $impact$.

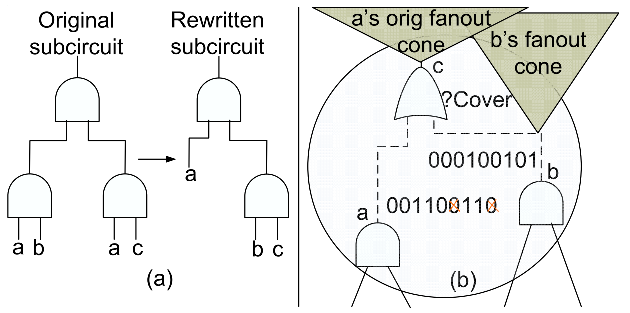

Fig. 11. (a) Rewriting a subcircuit to improve area. (b) Finding a candidate cover for node a.

$relODCmask(g, h) = 01000100\&01110110) = 01000100$. If $Perr = p$, then including faults on h itself, the impact of h is $5p/8 + 2p/8 = 7p/8$. In cases where some gates have higher intrinsic error probabilities than others, an average value of $Perr$ can be used.

This *impact* measure does not have to be used used with SiDeR, it can guide other techniques such as gate hardening, which rely on finding error-critical parts of a circuit [22]. We note however, that our measure does not take into account second-order effects, i.e., changes to the signatures and ODCmasks of other nodes, in cases where the additional logic actually *changes* the functionality of other nodes (through the use of ODCs).

Since AnSER maintains signatures and ODC information for each node, we can quickly find covers for resynthesis. Figure 11 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. Also the use of SAT-based verification allows for the use of approximate ODC computation (rather than exact) to identify candidates as well. We use a SAT solver (MiniSAT) to check equivalence by constructing miters along a cut in the fan-out cone of x between the original circuit and the new circuit with cover $f(x, y)$. For further details on SAT-based verification of logic optimizations see [21], [29].

B. Guided Local Rewriting

In this section, we demonstrate the use of AnSER to guide an external logic synthesis technique known as logic rewriting. Rewriting is a general technique that optimizes small subcircuits to obtain overall area improvements [17]. We optimize circuits simultaneously for SER and area by using AnSER to accept or reject rewrites.

The rewriting technique relies on the fact that different irredundant topologies corresponding to the same Boolean function can exhibit different SER characteristics. For instance, the circuit $\mathrm{AND}(A, \mathrm{AND}(B, C))$ is more reliable than the circuit $\mathrm{AND}(B, \mathrm{AND}(A, C))$ if $P[A = 0] > P[B = 0]$ since A will mask more errors than B in this case. Due to the heavy dependence on signal probability, enumerating such cases is difficult. This is precisely where AnSER's speed can aid in deciding between certain optimizations for a particular subcircuit.

The implementation of rewriting reported in [1], [17] first derives a 4-input cut for a selected node, defining a one-output subcircuit. Functionally-equivalent replacement candidates are found using look-up tables. To extend the algorithms described in [1] to improve SER and area, we rewrite 4-input subcircuits to improve their reliability. To ensure global SER improvement, we re-simulate the circuit and update SER estimates. Computational efficiency is achieved through fast incremental updates by AnSER. Further, we quickly prune candidate rewrites based on how they change the impact of the rewritten subcircuit. By extending the notion of impact to one-output subcircuits, we require only local computations.

Figure 11a illustrates two candidate rewrites. The original subcircuit with three gates can be rewritten with two gates. New nodal equivalences for the rewritten circuit can quickly be identified using structural hashing to further reducing area. In Figure 11a, we also observe that the contributions of the two equivalent subcircuits to the SER are different. The larger circuit, which has a redundant input a, allows for more logic masking.

C. Remarks

Without proper extensions, our resynthesis techniques may negatively affect delay and testability of a given circuit. Since SiDeR decreases the testability of nodes, more test patterns may be necessary for testing, or some nodes may become untestable. Fortunately, AnSER maintains testability and signature information for every node in the circuit. Indeed, for a node g, the $test_0(g)$ and $test_1(g)$ measures are an approximation of the random pattern testability of node g. Therefore, we can output both the testability and test vectors for any circuit node. Additionally, if we are given a set of test vectors to preserve, we can avoid node mergers that render a signal untestable by any of the given test vectors. Since we re-evaluate signatures and ODCmasks after each change in the circuit, the updated signatures and ODCmasks indicate the new test vectors for a particular node. If these vectors are not among the given test vectors then the change can be rejected.

The precise analysis of circuit delay requires technology mapping and interconnect lengths which are not available during technology-independent logic synthesis. If critical path information were available, the delay overhead could be decreased in SiDeR by prohibiting gate additions along those paths. For our rewriting technique, we could modify the objective function for selecting rewrites by requiring that modifications along critical paths to either maintain or decrease delay.

It is also possible to annotate SiDeR transformations on the netlist, allowing a downstream physical synthesis tool to undo optimizations when accurate timing analysis can be performed. Since fewer than 10% of gates and wires are timing-critical in heavily-optimized ICs, this "undo" functionality should be able to meet original timing constraints while also preserving the improvements in reliability achieved by SiDeR.

VII. Empirical Validation

We now report empirical results for SER analysis using AnSER and our two SER-aware synthesis techniques. The experiments were conducted on a 2.4 GHz AMD Athlon 4000+ workstation with 2GB of RAM. The algorithms were implemented in C++.

For validation purposes, we compare AnSER which computes SER under the TSA fault model with complete test-vector enumeration using the ATPG tool ATALANTA [15]. We provided ATALANTA with a list all of possible stuck-at (SA) faults in the circuit to generate tests in "diagnostic mode," which generates all test vectors for each fault. We used an intrinsic gate fault value of $Gerr = 1 \times 10^6$ on all faults. 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 fault probability as in Equation 4. While the exact computation can be performed only for small benchmarks, Table I suggests that our algorithm is accurate to about 3% for $2,048$ simulation vectors. More test vectors can be used if desired.

We isolate the effects of the two possible sources of inaccuracy: 1) sampling inaccuracy, and 2) inaccuracy due to approximate ODC computation. Sampling inaccuracy is due to the incomplete enumeration of the input space. Approximate ODCs computed using the algorithm from [21] incur inaccuracy due to mutual masking. When a fault is propagated through two reconvergent paths, they may cancel each other out. However, results in Table I indicate that most of the inaccuracy is due to sampling and not due to approximate ODCs. The last two columns of Table I, corresponding to exact ODC computation, show an average of 2.65% error. Therefore only 0.41% of the error is due to the approximate ODC computation. On the other hand, while enumerating the entire input space is intractable, our use of bit-parallel computation enables significantly more vectors to be sampled than other techniques [2], [23], [28] given the same amount of time.

To obtain accurate gate characterization information for the experiments, we adapted data from [23], where several gate types are analyzed in a 130nm, $1.2V_{DD}$ technology via SPICE simulations. We use an average SER value of 8×10^{-5} for all gates. However, the SER analyzers from [23], [27], [28] all report error rates that differ by orders of magnitude. SERA tends to report error rates on the order of 10^{-3} for 180nm technology nodes, and FASER reports error rates on the order of 10^{-5} for 100nm. Further, although our focus is logic masking, we approximate electrical masking by scaling our fault probabilities at nodes by a small derating factor to obtain trends similar to [23]. In Figure 12, we compare AnSER and SERD when computing SER for inverter chains of varying lengths. Since there is only one path in this circuit that is always sensitized, it helps us estimate the derating factor.

Circuit	No. gates	Runtime (s)			
		AnSER	SERD [23]	FASER [27]	[7]
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 II
Runtime comparisons with SER analyzers from [7], [23], [27].

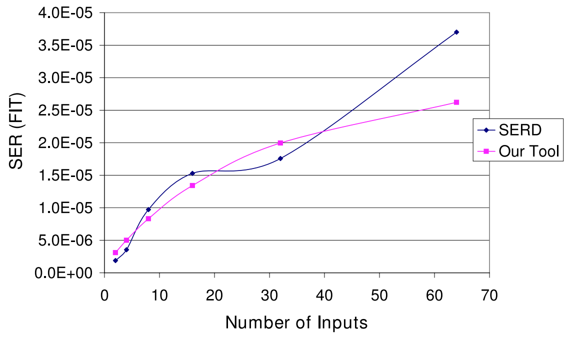

Fig. 12. Comparison of SER trends on inverter chains produced by SERD [23] and AnSER.

Table II compares AnSER with the prior art on ISCAS-85 benchmarks, using similar or identical host CPUs. While the runtimes in [7] include 50 runs, the runtimes in [23] are reported per input vector. Thus we multiply data from [23] by the number of vectors $(2,048)$ used there. Our runtimes appear better by several orders of magnitude. We believe that our faster run times are in large part due to the use of bit-parallel functional simulation to determine logic masking, which has a strong input-vector dependency. Most other works use fault simulation or symbolic methods.

Table III shows SER and runtime results for IWLS benchmarks that were evaluated when we implemented AnSER within the OAGear package [11]. [1] Note that our algorithm scales linearly in the size of the circuit, unlike the majority of prior algorithms. We assume a uniform input distribution in these experiments although AnSER is not limited to any particular input distribution. An input distribution supplied by a user, a sequential gate-level simulator, or a Verilog simulator can be used directly, even if it includes repeated vectors. In the latter case, all calculations based on signatures will remain correct. SER and runtime results with exact and approximate ODCs are shown on the ISCAS85 benchmarks in Table IV. Again, results show that approximate ODCs are sufficient for most benchmark circuits since the loss of accuracy due to ODC approximation is negligible.

[1] AnSER will be included as part of the OAGear package in the next release. OAGear can be downloaded from http://openedatools.si2.org/oagear/oa.html.

Circuit	No. gates	ATALANTA	AnSER	% Error	AnSER Exact-ODC	% Error
c17	13	6.96E-7	6.96E-7	0.01	6.96E-7	0.01
majority	21	6.25E-6	6.63E-6	6.05	6.57E-6	4.87
decod	25	2.60E-5	2.62E-5	0.83	2.60E-5	0.83
b1	25	1.28E-5	1.31E-5	2.81	1.27E-5	0.78
pm1	68	2.86E-5	3.00E-5	4.70	2.97E-5	3.5
tcon	80	5.30E-5	5.39E-5	1.67	5.35E-5	0.94
x2	86	3.78E-5	3.87E-5	2.38	3.93E-5	3.97
z4ml	92	5.29E-5	5.37E-5	1.50	5.41 E-5	2.20
parity	111	7.60E-5	7.69E-5	1.24	7.71E-5	1.45
pcle	115	5.38E-5	5.34E-5	0.75	5.35E-5	0.56
pcler8	140	7.06E-5	7.24E-5	2.52	7.23E-5	2.41
mux	188	1.58E-5	1.38E-5	12.54	1.63E-5	3.16
Average				3.06		2.65

TABLE I
COMPARISON SER (FIT) DATA FOR ANSER AND ATALANTA.

Circuit	No. Gates	SER (FIT)	Runtime (s)
pci_conf_cyc_addr_dec	97	4.89E-3	0.23
steppermotordrive	226	8.00E-3	0.27
ss_pcm	470	1.68E-2	0.3
usb_phy	546	1.53E-2	0.28
sasc	549	2.10E-2	0.26
simple_spi	821	2.50E-2	0.3
i2c	1142	2.7E-2	0.34
pci_spoci_ctrl	1267	0.029	0.342
des_area	3132	0.019	0.782
spi	3227	0.118	0.68
systemcdes	3322	0.127	0.55
tv80	7161	0.104	0.91
systemcaes	7959	0.267	0.97
mem_ctrl	11440	0.494	1.36
ac97_ctrl	11855	0.409	1.38
usb_funct	12808	0.390	1.42
pci_bridge32	16816	0.656	1.78
aes_core	20795	0.550	2.1
wb_conmax	29034	1.030	4.18
ethernet	46771	1.480	5.77
des_perf	98341	3.620	9.34
vga_lcd	124031	4.800	11.7

TABLE III
SER (IN FITS) AND RUNTIME FOR ANSER ON THE IWLS 2005 BENCHMARKS.

Circuits	No. gates	SER (FIT)	Runtime (s)	SER (FIT)	Runtime (s)
alu4	740	1.13E-2	0.227	1.19E-2	0.004
b9	14	4.67E-3	0.007	4.69E-3	0.005
b1	114	6.79E-3	0.050	6.69E-3	0.000
C1355	536	1.93E-2	2.010	1.93E-3	0.034
C3540	1055	3.06E-2	0.409	3.07E-2	0.080
C432	215	5.70E-3	0.056	5.71E-3	0.016
C499	432	1.75E-2	0.291	1.71E-2	0.260
C880	341	1.50E-2	0.54	1.51E-2	0.23
cordic	84	9.43E-2	0.007	9.43E-2	.004
dalu	1387	2.18E-2	0.535	2.17E-2	0.225
des	4252	2.04E-1	5.283	2.03E-1	4.87
frg2	1228	3.61E-1	0.217	3.65E-1	0.169
i10	2824	1.03E-1	1.063	1.04E-1	0.315
i9	952	5.07E-2	2.237	5.06E-2	2.044

TABLE IV
SER EVALUATION OF VARIOUS BENCHMARKS WITH EXACT AND APPROXIMATE ODCS.

Circuit	No. gates	No. cycles	Runtime(s)	
			MARS-S	AnSER
s208	112	10	1000	1
s298	133	10	6900	0
s444	181	10	365	4
s526	214	5	551	11
s1196	547	5	68	8
s1238	526	4	70	8

TABLE V
COMPARISON OF MULTI-CYCLE SIMULATION RUNTIMES.

Table V compares the multi-cycle simulation runtimes of of AnSER with this of MARS-S, the sequential circuit SER analyzer from [19]. MARS-S employs symbolic simulations using a BDD/ADD-based framework to compute steady-state probability distributions while we use signature-based bit-parallel functional simulations. The number of cycles needed to reach steady state is also listed in the table. Table VI shows the results of SER analysis on sequential circuits from the ISCAS-89 benchmark suite under time-frame expansion. The listed runtimes in Table VI are for processing signatures and ODCs on 10 frames. These results indicate that the SER obtained by considering only one time frame is 62% higher than the 2-frame SER. After this point, increasing the number of frames has little effect on the SER. This indicates that most faults, if at all propagated, are usually observable at primary outputs of the current cycle. This result is supported by observations in [10]. In other words, flip-flops propagate few errors to the outputs in later cycles due to sequential circuit masking. The latched errors tend to quickly dissipate with the number of cycles leaving the SER for multiple-cycle analysis close to the error rate of the current cycle's primary outputs.

Table VII shows improvements in SER and area overhead obtained by SiDeR. The first set of results are for exact covers, i.e., covers that do not consider ODCs. The second set uses ODCs to increase the number of candidates as well as to maintain testability. Since the use of ODCs results in candidate-target pairs that are not identical, faults at the output of either gate can still be propagated in most cases. In both cases, AND/OR gates are used according to the covering relationship. For exact covers, we see an average of 29.1% SER improvement with only 5% area overhead. The improvements for the ODC covers are 39.8% with area overhead of 13.1%.

Table VIII illustrates the use of AnSER to guide the local rewriting implementation in the ABC logic-synthesis package [1]. AnSER calculates the global SER 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, in the case of `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, these results show that carefully guided logic transformations can eliminate this problem.

Table IX shows the results of combining SiDeR and local

Circuit	No. gates	Runtime (s)	SER for n time-frames					
			n=0	n=1	n=2	n=3	n=4	n=5
s208	112	9	2.40E-3	2.34E-3	2.34E-3	2.33E-3	2.32E-3	2.33E-3
s298	133	9	2.97E-3	2.75E-3	2.69E-3	2.67E-3	2.65E-3	2.62E-3
s400	180	14	4.24E-3	3.00E-3	2.38E-3	2.23E-3	2.23E-3	2.05E-3
s444	181	14	4.69E-3	3.06E-3	2.43E-3	2.18E-3	2.02E-3	1.98E-3
s526	214	9	3.87E-3	2.97E-3	2.65E-3	2.52E-3	2.46E-3	2.44E-3
s1196	547	18	6.35E-3	3.87E-3	3.71E-3	3.68E-3	4.05E-3	3.89E-3
s1238	526	14	6.09E-3	3.54E-3	3.42E-3	3.47E-3	3.72E-3	3.62E-3
s1488	659	5	1.02E-1	1.11E-2	1.03E-2	1.06E-2	1.15E-2	1.07E-2
s1423	731	47	1.43E-2	8.48E-3	5.08E-3	3.47E-3	2.78E-3	2.54E-3
s9234	746	4	1.31E-2	1.24E-2	1.22E-2	1.18E-2	1.07E-2	9.78E-2
s13207	1090	15	3.07E-2	2.66E-2	3.14E-2	3.62E-2	3.61E-2	4.39E-2

TABLE VI
THE CHANGE IN SER FOR SEQUENTIAL CIRCUITS WITH INCREASING NUMBER OF TIME FRAMES.

Circuit	With exact covers		With approx covers	
	% SER improvement	% Area overhead	% SER improvement	% Area overhead
cordic	1.7	1.2	27.3	45.2
b9	18.1	14.9	30.7	31.6
C432	37.6	14.0	38.7	14.9
C880	9.6	0.9	13.1	2.3
C499	1.0	3.2	32.2	20.6
C1908	5.9	9.0	32.4	24.1
C1355	25.3	9.0	30.7	8.6
alu4	55.9	0.9	55.9	1.6
i9	65.4	6.6	65.4	6.6
C3540	31.1	2.2	49.4	3.6
dalu	74.3	1.2	74.3	1.2
i10	40.4	5.4	40.4	5.6
des	11.4	2.9	26.7	4.4
Avg	**29.1**	**5.5**	**39.8**	**13.1**

TABLE VII
IMPROVEMENTS IN SER OBTAINED BY SIDER.

Technique	% SER improvement	% Area overhead
SEROnly [2]	14.9	8.2
JointOpt [2]	6.8	1.3
IndirImply [3]	9.4	6.7
BackJustify [3]	14.4	8.6
DirImply [3]	15.7	8.64
PartialMask [20]	82.7	107.4
DomValue [20]	77.3	81.1
SiDeR	29.1	5.5
SiDeR-ODC	39.8	13.1
Rewriting	13.7	-2.3
Combined	68.5	26.3

TABLE X
COMPARISON OF THE VARIOUS SER IMPROVEMENT TECHNIQUES.

Circuits	No. rewrites	% SER improvement	% Area improvement	Time (s)
alu4	13	29.3	0.5	24.5
b1	0	0.0	0.0	0.2
b9	8	6.8	0.9	0.3
C1355	97	1.2	9.0	37.6
C3540	23	5.8	0.9	51.5
C432	68	5.5	1.4	12.1
C499	37	0.0	0.5	13.0
C880	7	0.2	0.0	5.4
cordic	5	1.2	1.2	0.5
dalu	58	24.0	3.2	35.0
des	282	11.2	0.1	12.3
frg2	96	27.9	2.0	8.9
i10	143	5.0	0.6	16.7
i9	83	31.4	11.7	35.3
Avg		**13.7**	**2.3**	**18.1**

TABLE VIII
IMPROVEMENTS IN SER AND AREA WITH LOCAL REWRITING.

Circuit	% SER improvement	% Area overhead
alu4	95.33	55.41
b1	8.08	14.29
b9	19.88	25.44
C1355	99.49	19.40
C3540	96.02	39.72
C432	96.81	22.79
C499	86.74	14.58
C880	59.58	24.93
cordic	58.34	33.33
dalu	92.68	41.17
des	40.41	-1.69
frg2	46.42	27.85
i10	80.67	2.16
i9	78.05	49.26
Average	68.46	26.33

TABLE IX
IMPROVEMENTS IN SER BY COMBINING OF REWRITING AND SIDER.

rewriting. In this experiment, we first used SiDeR followed by two passes of rewriting (in area-unconstrained and area-constrained modes) to improve both area and SER. This particular combination of the two techniques yields 68% improvement in SER with 26% area overhead. The improvements seen are not necessarily additive as one optimization may change the starting point and available options for the other. Further, different interleavings of the two optimizations can provide different results.

Table X compares various SER-aware logic synthesis optimizations. The first two techniques are from [2], which uses ATPG-based rewiring. *SEROnly* refers to optimizing only the SER, while *JointOpt* refers to joint optimization for SER, area, power and delay. The next three techniques are from [3]. Here, logic implication analysis is used to identify redundant wires. *IndirImply*, *BackJustify* and *DirImply* are techniques that incorporate indirect implications, backwards justification, and direct implications respectively. The next two techniques from [20] are variants of partial fault masking. Nodes identified as susceptible are triplicated in the *PartialMask* technique, while these nodes are only duplicated in *DomValue*. The final four techniques are ours —the basic SiDeR technique, SiDeR augmented with ODCs, guided local rewriting, and combined SiDeR/rewriting.

Generally, these results show that guiding logic optimization techniques such as those of [2], [3] incur less overhead than explicit replication [20]. Further, our results indicate that SiDeR is able to identify more inherent redundancy than implication-based analysis [3]. A possible explanation for this is that implication analysis is restricted to certain types of implications and certain parts of the circuit. Further, it is difficult to incorporate ODCs and SDCs (satisfiability don't cares) in implication analysis. Our signature-based analysis on the other hand, implicitly incorporates SDCs and we explicitly compute ODCmasks to incorporate don't cares in our synthesis techniques.

VIII. Conclusions

We have presented an SER-aware design framework which includes a logic-level fault model, SER evaluation algorithms, and logic synthesis techniques. AnSER, our technology-independent SER analyzer, accurately evaluates the logic masking in combinational and sequential logic circuits. It achieves very high speed (2-3 orders of magnitude faster than previous methods) through the efficient use of node signatures and ODC masks. We also proposed a new, SER-aware resynthesis strategy, SiDeR, which manipulates node signatures to find redundancies within the circuit. Then, with the addition of a few fault-masking gates, SiDeR protects the fan-in cones of vulnerable nodes. On average, SiDeR decreases SER by 40% with only 13% area overhead. Finally, we successfully applied AnSER to local rewriting, and showed that this approach simultaneously improves area and SER. Combining the two techniques gives a 68% SER improvement.

Acknowledgment. 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, Y. Makris, Y-S. Yang, A. Veneris, "Seamless Integration of SER in Rewiring-Based Design Space Exploration," *ITC 2006*, pp. 1-9.

[3] S. Almukhaizim, Y. Makris, "Soft Error Mitigation Through Selective Addition of Functionally Redundant Wires, Transactions on Reliabilty," March 2008, vol. 57, no. 1, pp. 23-31.

[4] M. Bushnell, V. Agrawal, *Essentials of Electronic Testing*, Kluwer, 2000.

[5] S.-C. Chang, L.P.P.P. Van Ginneken, M. Marek-Sadowska," Circuit Optimization by Rewiring," *IEEE trans. on Computers*, vol. 48, no. 9, September 1999, pp. 962-970.

[6] H. Cho, S.-W. Jeong, F. Somenzi, C. Pixley, "Synchronizing Sequences and Symbolic Traversal Techniques in Test Generation,"*Journ. of Electronic Testing: Theory and Applications*, vol. 4, 1993, pp. 1931.

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

[8] L.A. Entrena, J.A. Espejo, E. Olias, J. Uceda, "Timing Optimization by an Improved Redundancy Addition and Removal Technique," *EURO-DAC*, 1996, pp. 342-347.

[9] G. D. Hachtel, E. Macii, A. Pardo, F. Somenzi, "Markovian Analysis of Large Finite State Machines," *IEEE Trans. on CAD*, vol. 15, December 1996, pp. 1479-1493.

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

[11] S. Krishnaswamy, S. M. Plaza, I. L. Markov, and J. P. Hayes, "AnSER: A Lightweight Reliability Evaluator for use in Logic Synthesis," IWLS, San Diego, CA, 2007. http://www.iwls.org/challenge-2008/history2.html

[12] 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.

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

[14] W. Kunz, "Multi-level Logic Optimization by Implication Analysis," *ICCAD 1994*, pp. 6-13.

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

[16] E. S. Millan, L. Entrena, J.A. Espejo, "On the Optimization Power of Redundancy Addition and Removal Techniques for Sequential Circuits," *Proc. ICCAD*, 2001, pp. 91-94.

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

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

[19] N. Miskov-Zivanov, D. Marculescu, "Soft Error Rate Analysis for Sequential Circuits," *DATE 2007*, pp. 1436-1441.

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

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

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

[23] R. Rao, K. Chopra, D. Blaauw, D. Sylvester, "An Efficient Static Algorithm for Computing the Soft Error Rates of Combinational Circuits," *DATE 2006*, pp. 164-169.

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

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

[26] A. Veneris, M. Abadir, "Design Rewiring using ATPG," *IEEE Trans. on CAD*, 2002, vol. 21, no. 12, pp. 1469-1479.

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

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

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