Soft Error Mitigation Through Selective Addition of Functionally Redundant Wires

Sobeeh Almukhaizim, *Member, IEEE*, and Yiorgos Makris, *Member, IEEE*

***Abstract*—We introduce a logic-level soft error mitigation methodology for combinational circuits. The proposed method exploits the existence of logic implications in a design, and is based on selective addition of pertinent functionally redundant wires to the circuit. We demonstrate that the addition of functionally redundant wires reduces the probability that a Single-Event Transient (SET) error will reach a primary output, and, by extension, the Soft Error Rate (SER) of the circuit. We discuss three methods for identifying candidate functionally redundant wires, and we outline the necessary conditions for adding them to the circuit. We then present an algorithm that assesses the SET sensitization probability reduction achieved by candidate functionally redundant wires, and selects an appropriate subset that, when added to the design, minimizes its SER. Experimental results on ISCAS'89 benchmark circuits demonstrate that the proposed soft error mitigation methodology yields a significant SER reduction at the expense of commensurate hardware, power, and delay overhead.**

***Index Terms*—Logic implications, single-event transient, soft error rate, soft error sensitization probability.**

ACRONYM[1]

SER	Soft Error Rate
SET	Single-Event Transient
B.J.	Backward Justification
D.I.	Direct Implications
I.I.	Indirect Implications

NOTATION

$G_1 \ldots G_{10}$	logic gates
u, v	logic values
W_s	source wire of an implication
W_t	target wire of an implication
$a, b, \ldots, h$	circuit inputs
N	number of SET
M	number of random input vectors
k_i	number of implications that mask SET i
R_{SET}	rate of occurrence of a SET at a gate
P_{sens}	probability of a SET reaching an output
P_{latch}	probability that a SET is latched in a storage element

I. INTRODUCTION

WHEN high-energy neutrons or alpha particles strike a sensitive region in a semiconductor, they generate a Single-Event Transient (SET), in the form of an unexpected current pulse of a short duration. Under certain circumstances, the latter may be misinterpreted by the circuit as a valid signal, and result in an incorrect state and/or output, thus producing a *soft error*. Such soft errors, which only distort the data processed by the circuit but make no damage to the hardware itself, are emerging as a serious threat to the reliable operation of logic circuits. Historically, soft errors have been of great concern in memories, and various mitigation solutions have been developed [1]. Memories occupy a large area of silicon, and comprise storage elements that are much more susceptible to particle strikes than combinational logic [2], where SET are frequently masked before reaching an output or a storage element [3]. However, technological trends such as faster clock rates, smaller device sizes, lower supply voltages, and shallower logic depths are drastically reducing SET masking, and significantly increasing the occurrence of soft errors in combinational logic [1]. Therefore, mitigation methods to reduce the Soft Error Rate (SER) in combinational logic are vital to the reliable operation of integrated circuits.

To this end, this paper contributes a soft error mitigation method which is based on the selective addition of functionally redundant wires to the combinational logic of a circuit. The proposed method builds upon the concept of *logic implications*. These implications reflect fine-grained invariant relations between logic-level signals, and reveal opportunities for adding functionally redundant wires to the circuit, in order to reduce its susceptibility to soft errors. The fundamental mechanism through which soft errors are mitigated is quite simple: if a SET distorts a signal, then the existence of appropriately selected functionally redundant wires can prevent the distorted signal from propagating to an output or a storage element, and resulting in a soft error. As a side effect, however, the addition of functionally redundant wires not only incurs hardware, power, and delay overhead, but also introduces new locations wherein SET may occur. Therefore, a systematic mechanism for identifying & judiciously adding functionally redundant wires to the circuit is required. For this purpose, we employ three distinct methods for identifying logic implications in a combinational circuit, and we discuss the construction of

Manuscript received December 9, 2006; revised July 4, 2007; accepted July 9, 2007. Associate Editor: J.-C. Lu.

S. Almukhaizim is with the Department of Computer Engineering at Kuwait University, Kuwait (e-mail: sobeeh@eng.kuniv.edu.kw).

Y. Makris is with the Departments of Electrical Engineering & Computer Science at Yale University, New Haven, CT 06511 USA (e-mail: yiorgos.makris@yale.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TR.2008.916877

[1]The singular and plural of an acronym are always spelled the same.

functionally redundant wires to realize these implications. Subsequently, we propose a selection algorithm which assesses the benefits of candidate functionally redundant wires, and selects a cost-effective subset among them to reduce the SER of the circuit.

Unlike previous soft error mitigation methods, which operate at the circuit level, and are only applicable once the circuit has been mapped to a specific technology, the uniqueness of the proposed method is that it operates at the logic level, and can therefore be applied much earlier in the design cycle in a technology-independent fashion. Moreover, the mechanisms through which soft errors are mitigated at the logic level are orthogonal to those at the circuit level; hence, the logic-level method proposed herein not only provides a better starting point, but may also be applied synergistically with the current state of the art in circuit-level soft error mitigation.

The remainder of the paper is organized as follows. In Section II, we review related work in soft error mitigation for combinational logic. In Section III, we discuss how the addition of a functionally redundant wire can increase the probability of masking SET in a circuit. Then, in Section IV, we describe three methods for identifying functionally redundant wires for potential addition to the circuit. In Section V, we examine the conditions by which such an addition should abide. In Section VI, we present an algorithm for selecting & adding a cost-effective subset of functionally redundant wires to the circuit to reduce its SER. Experimental results on ISCAS'89 benchmark circuits are provided in Section VII, demonstrating that the proposed algorithm achieves significant reduction in the SER of a circuit, commensurate with the incurred hardware, power, and delay overhead.

II. Prior Work

The SER of a combinational circuit is proportional to three factors [1], [4]: the rate of SET occurrence at a logic gate (R_{SET}), the probability of a SET arriving at a storage element during its latching window (P_{latch}), and the probability of a SET propagating to an output or a storage element through a sensitized path (P_{sens}). To reduce the SER of a circuit, previous soft error mitigation methods have focused on the first two factors.

Soft error mitigation via $\mathbf{R_{SET}}$ **reduction** is based on circuit-level design modifications, wherein individual transistor characteristics are perturbed to reduce the sensitivity of logic gates to SET. Specifically, such methods resize the (W)idth/(L)ength ratios of the transistors in a select set of gates to increase their immunity to SET; and, by extension, reduce R_{SET}, and the SER of the circuit. SER estimation & assessment of gate susceptibility to soft errors is performed either through fault injection & simulation, wherein the SET masking factors are evaluated separately [4]–[6]; or through symbolic representation, wherein all SET masking factors are evaluated in a unified approach [7]–[9]. In either way, gate resizing is performed for the most susceptible logic gates, i.e. the ones that contribute the most to the SER of the design [10]–[13].

Soft error mitigation via $\mathbf{P_{latch}}$ **reduction** is based on redesigning the storage elements of the circuit to prevent latching of SET occurring in flip-flops [14], [15], combinational logic [16]–[18], or both [19]–[21]. Specifically, such methods take advantage of the temporary nature of SET, and tolerate them via fine-grained time redundancy [16]–[18]. In this case, the flip-flop inputs are sampled multiple times within the slack time available for each output of the circuit; and a majority voter selects the winner of the sampled values, which is subsequently stored in each flip-flop. This reduces the SET latching window; and, by extension, the P_{latch}, and the SER of the circuit. In an effort to reduce the incurred cost, several methods that utilize existing test & debug system resources to implement time redundancy-based soft error mitigation have been recently proposed [20], [21].

In contrast to these circuit-level solutions, the method proposed herein mitigates soft errors at the logic level via $\mathbf{P_{sens}}$ **reduction**, which offers several advantages. First, the proposed method is technology-independent; thus, it enables design modifications for SER reduction that are equally effective, independent of the technology to which the circuit will be eventually mapped. Second, it allows SER to be considered as a design objective earlier in the design cycle, when only the logic-level netlist is available. Third, it targets an SER contributing factor that is not addressed by the previously proposed circuit-level methods, namely P_{sens}; therefore, it naturally complements, and can be combined with soft error mitigation methods that aim at reducing R_{SET}, and P_{latch}.

III. Basic Principle

The proposed method is based on the existence of implications, i.e. fine-grained invariant relations between pairs of wires in a circuit. An implication from a source wire to a target wire indicates that a value assignment at the source wire forces a consistent value assignment at the target wire. Such forced relations provide a source of invariance that can be used to mask SET. Suppose that a SET distorts the target wire by changing the value of a gate on the implication path between the source wire, and the target wire. Then, if the source wire is added as an input to the gates driven by the erroneous target wire, the effect of the distorted value can be masked, and the output of the circuit will still be correct. Addition of such wires, however, is possible only if the function realized is preserved, i.e. if the added wire is functionally redundant.

Fig. 1 illustrates the idea of utilizing logic implications to add redundant wires to the circuit. Let $(W_s, u) \Rightarrow (W_t, v)$ denote an implication, i.e. the fact that a value of u on the source wire W_s will cause a value of v on the target wire W_t, where $u, v \in \{0,1\}$. In the circuit of Fig. 1, for example, one such implication would be $(e,1) \Rightarrow (G_8,0)$, because if $e=1$, then $G_3 = 1$, and $G_8 = 0$. Consider the case where $e=1$, and $G_6 = G_7 = 0$. Given that $e=1$, and $(e,1) \Rightarrow (G_8,0)$, we expect that $G_8 = 0$. If a SET flips the value of any gate on the implication path (comprising G_3, and G_8, in our case), then the output of G_8 will obtain an erroneous value of 1. If $h=1$, this error will propagate through G_{10} to O_1. However, if we add the *functionally redundant* dotted wire & inverter from e to G_{10} (realizing the implication $(e,1) \Rightarrow (G_8,0)$), then the SET will be masked at G_{10} before reaching O_1. The addition of the redundant wire introduces a new location where SET may appear, and propagate to the output. This is the case if a SET affects the

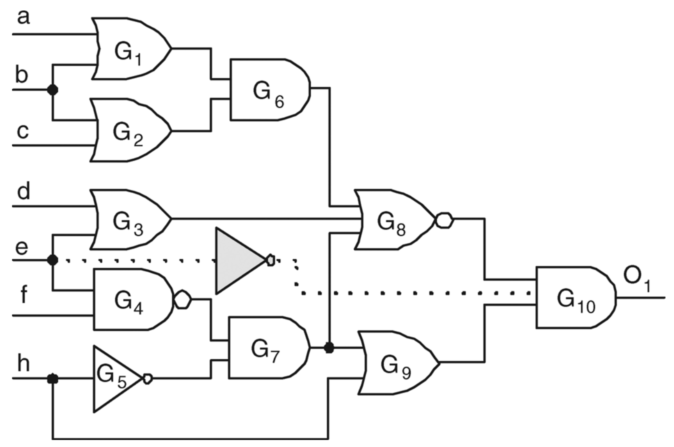

Fig. 1. Example circuit.

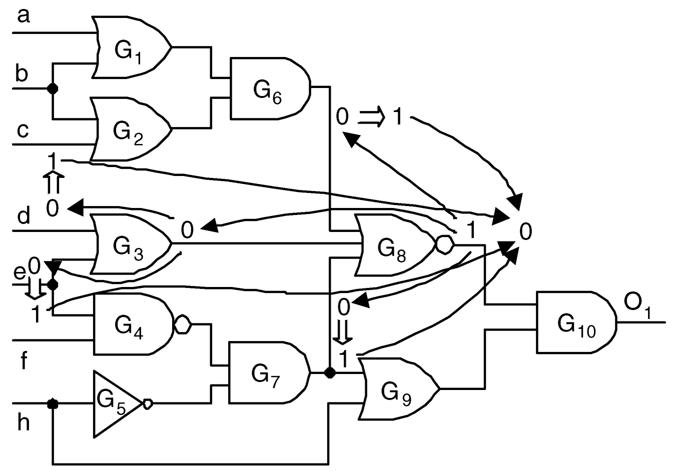

Fig. 2. Identification of implications using backward justification.

added inverter while $e = 0$, and $G_8 = G_9 = 1$. If the overall sensitization probability, however, of all the potential SET in the circuit is reduced, then the corresponding redundant wire effectively reduces the SER. In other words, the reduction to the probability of sensitization, when a redundant wire is added, represents how critical the redundant wire is to the reliability of the system as it measures the improvement in its reliability against soft errors.

Essentially, a SET that distorts the value of a gate on an implication path can be masked by adding a wire from the source of the implication to the gates driven by the target wire. When such a wire is added to the logic circuit, the SER of the circuit is reduced, and therefore the operation of the circuit in the presence of SET is more reliable. To develop this basic principle into a soft error mitigation methodology, however, the following **three key questions** need to be answered: **i)** How do we identify candidate functionally redundant wires in a circuit? **ii)** Under what conditions can these wires be added to the circuit? **iii)** How do we select a cost-effective subset among the possibly large number of candidate wires?

IV. Identification of Candidate Redundant Wires

In this section, we discuss how to find candidate redundant wires that can be added to a logic circuit to mask a particular SET. Candidate functionally redundant wires are identified when there exists an implication $(W_s, u) \Rightarrow (W_t, v)$. Given a specific target wire & value pair, (W_t, u), several methods can be used to find implicating source wires & values, (W_s, u). Three such methods, varying in computational complexity & effectiveness, are discussed in the following subsections.

A. Backward Justification

Backward justification [22] aims at identifying implications $(W_s, u) \Rightarrow (W_t, v)$ with a source wire W_s in the cone of logic driving the target wire W_t. This is done by first setting W_t to $\overline{v}$. Then, an attempt is made to justify the value assignment of the inputs of the gate driving W_t, and every other gate in its logic cone, based on the connectivity of the circuit. Justification of gates in the logic cone driving W_t is feasible if there exists a single possible value for the unjustified gates. Once backward justification is completed, a wire W_s in the cone of logic driving W_t may be assigned a value, say $\overline{u}$, that is represented using the logic implication

$$(W_t, \overline{v}) \Rightarrow (W_s, \overline{u}). \tag{1}$$

Or equivalently, by taking the contrapositive,

$$(W_s, u) \Rightarrow (W_t, v), \tag{2}$$

which represents the implication of W_s on W_t.

All wires justified through the backward justification procedure constitute a possible source point for a functionally redundant wire. In this case, the complexity of identifying the implications is low as it only requires a single pass from the target wire to the primary inputs.

As an example, consider the circuit of Fig. 2; and assume that we want to identify source wires, and values that implicate $(G_8, 0)$. The output of G_8 is set to 1, and its inputs, G_3, G_6, and G_7, are justified to 0 because G_8 is a NOR gate. Next, the output of G_3 is set to 0, and its inputs, d, and e, are both justified to 0. Therefore, $(G_8, 1) \Rightarrow ((d, 0), (e, 0), (G_3, 0), (G_6, 0), (G_7, 0))$, which is equivalent to the following implications, summarized in the fourth column of Table I: $(d, 1) \Rightarrow (G_8, 0), (e, 1) \Rightarrow (G_8, 0), (G_3, 1) \Rightarrow (G_8, 0)$,$(G_6, 1) \Rightarrow (G_8, 0)$, and $(G_7, 1) \Rightarrow (G_8, 0)$. Under certain conditions discussed in Section V, the source wire of any of these implications can be connected as an additional input to gates driven by the target wire G_8 (G_{10} in this case).

B. Direct Implications

Direct implications are identified by simply evaluating a gate with a given combination of value assignments at its inputs & output, and propagating the signal values according to the connectivity of the circuit. The justification process while finding direct implications is similar to the justification process while performing backward justification. In both cases, justification is feasible if there exists a single possible value for an unjustified gate. However, direct implications are implications not only from the cone of logic driving the target wire but also from the rest of the circuit. To find all direct implications in a circuit, backward justification, and forward justification are performed iteratively until every unjustified gate is either justified, or there

TABLE I
IMPLICATIONS IDENTIFIED FOR $G_8 = 0$ IN THE CIRCUIT OF FIG. 2 USING BACKWARD JUSTIFICATION (B.J.), DIRECT IMPLICATION (D.I.), AND INDIRECT IMPLICATION (I.I.)

Source	Implication	Inverted	B. J.	D. I.	I. I.
G_6	$1 \Rightarrow 0$	Yes	$\surd$	$\surd$	$\surd$
G_3	$1 \Rightarrow 0$	Yes	$\surd$	$\surd$	$\surd$
d	$1 \Rightarrow 0$	Yes	$\surd$	$\surd$	$\surd$
e	$1 \Rightarrow 0$	Yes	$\surd$	$\surd$	$\surd$
G_7	$1 \Rightarrow 0$	Yes	$\surd$	$\surd$	$\surd$
h	$0 \Rightarrow 0$	No	-	$\surd$	$\surd$
G_4	$0 \Rightarrow 0$	No	-	$\surd$	$\surd$
G_5	$1 \Rightarrow 0$	Yes	-	$\surd$	$\surd$
G_9	$0 \Rightarrow 0$	No	-	$\surd$	$\surd$
G_{10}	$0 \Rightarrow 0$	No	-	$\surd$	$\surd$
b	$1 \Rightarrow 0$	Yes	-	-	$\surd$

exist more than one possible justification for it. In every iteration, all unjustified gates that have a new assignment to their inputs or output are evaluated for further possible justification. A well-known example of direct implications is the implication procedure of FAN[2] [23].

As an example, consider again the circuit of Fig. 2. After backward justification is performed with the output of G_8 set to 1, we find that d, e, G_3, G_6, and G_7 are all justified to 0. At this point, backward justification terminates because there are no more gates that can be justified. However, forward justification can be performed, yielding $G_4 = 1$, because $e = 0$. Thus, a new assignment is made to the input of G_7, which is now added to the list of unjustified gates. In the next iteration, backward justification is performed on G_7, yielding $G_5 = 0$, because $G_7 = 0$, and $G_4 = 1$. In addition, $h = 1$ because $G_5 = 0$. Finally, forward justification, using the assignment of $h = 1$, yields $G_9 = 1$, and $G_{10} = 1$; and the direct implication identification procedure terminates. Hence, 5 additional implications are identified for $G_8 = 0$, as shown in the fifth column of Table I. This results in more candidate wires for addition, at the cost of multiple backward, and forward justification passes.

C. Indirect Implications

The direct implication identification procedure covers the straightforward case where there is only a single possible value that justifies an unjustified gate. A gate is considered unjustified if the current assignments of its inputs do not justify the output value of the gate. For example, an AND gate that has an output value of 0 with none of its inputs assigned a value of 0 is an unjustified gate. In this case, there exist multiple possible input value combinations for justifying the gate. As a result, the direct implication identification procedure will stop. However, if all justifications of the unjustified gate yield a common implication, then this implication holds true regardless of the actual justification. This type of implication is called an *indirect implication*.

Several methods have been proposed in the literature for identification of indirect implications, which is computationally much harder than identification of only direct implications. The ATPG method presented in [24] is able to derive a few indirect implications in addition to direct implications. The main idea is to identify the implications through *Learning*, which was first introduced in [24], [25], and further developed in [26]. Learning is defined in [27] to mean the temporary injection of logic values at certain signals in the circuit, in order to examine their logical consequences. Techniques such as Extended Backward Learning [28], and Recursive Learning [27] have also been proposed to identify more implications in a logic circuit. The implications obtained through Extended Backward Learning are a strict subset of the implications obtained using Recursive Learning, because of the generalized unjustified gate definition used in [27]. In general, Recursive Learning [27] with unlimited recursion depth is the most powerful method, because it is able to identify all direct, and indirect implications for a given target wire, and value assignment. However, its time complexity is exponential in the recursion depth. Nevertheless, experiments from [29] indicate that a small depth is usually sufficient to identify most of the implications that exist in a realistic circuit.

As an example of indirect implications, consider G_6 in the circuit of Fig. 2. Given a value assignment of 0 at G_6, backward justification is unable to identify an implication because either $G_1 = 0$, or $G_2 = 0$ constitute a possible justification for $G_6 = 0$. Therefore, neither backward justification nor the direct implication identification procedure are able to identify additional implications. However, a closer examination reveals that $b = 0$ regardless of whether $G_1 = 0$ or $G_2 = 0$. In this case, the additional implication $(b, 1) \Rightarrow (G_8, 0)$ is an indirect implication that can only be found using a method that identifies common implications in all the justification scenarios, such as Recursive Learning.

The implications found by the indirect implication identification procedure for $G_8 = 0$ are summarized in the sixth column of Table I, and compared to the implications found by the previous two methods. Backward justification (B.J.) identifies five implications, direct implications (D.I.) identifies an additional five, and indirect implications (I.I.) identifies one additional implication. This example indicates that the indirect implication identification procedure is able to find more implications in the circuit than backward justification, and the direct implication identification procedure. However, the benefit of indirect implications seems marginal because the direct implication identification procedure finds almost all implications of $G_8 = 0$ in the circuit. To disperse this possibly misleading observation, we provide a second example where the direct implication identification procedure fails to identify any implications in the circuit, while the indirect implication identification procedure identifies all relevant implications.

Consider the circuit of Fig. 3, which is a modified version of the circuit of Fig. 1, where b is replaced with G_8, and the inversion is removed from G_8 then added to G_6. Assume that we would like to identify source wires that implicate $G_6 = 0$. Therefore, we first assign 1 to G_6, and perform backward justification. In this case, backward justification fails to justify the inputs to G_6, and no implications are identified. Next, we perform forward justification, which also fails to identify any implications. Therefore, both backward justification, and the direct implication identification procedure both fail to identify any implications. However, further analysis of the structure in the shaded area identifies the indirect implication $(G_6, 1) \Rightarrow (G_8, 0)$. As

[2]FAN is a fanout-oriented test pattern generation algorithm.

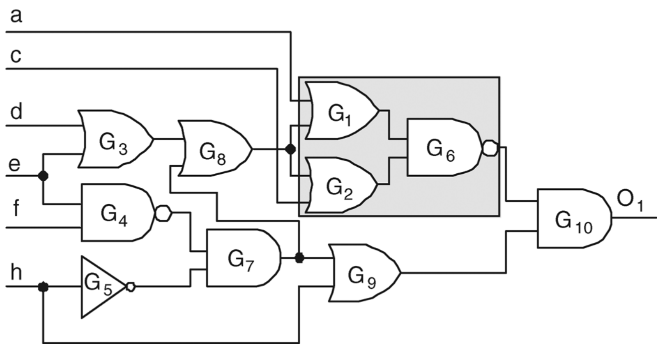

Fig. 3. Modified example circuit.

TABLE II
IMPLICATIONS IDENTIFIED FOR $G_6 = 0$ IN THE CIRCUIT OF FIG. 3 USING BACKWARD JUSTIFICATION (B.J.), DIRECT IMPLICATION (D.I.), AND INDIRECT IMPLICATION (I.I.)

Source	Implication	Inverted	B. J.	D. I.	I. I.
G_8	$1 \Rightarrow 0$	Yes	-	-	$\surd$
G_3	$1 \Rightarrow 0$	Yes	-	-	$\surd$
d	$1 \Rightarrow 0$	Yes	-	-	$\surd$
e	$1 \Rightarrow 0$	Yes	-	-	$\surd$
G_7	$1 \Rightarrow 0$	Yes	-	-	$\surd$
G_4	$0 \Rightarrow 0$	No	-	-	$\surd$
G_5	$1 \Rightarrow 0$	Yes	-	-	$\surd$
h	$0 \Rightarrow 0$	No	-	-	$\surd$
G_9	$0 \Rightarrow 0$	No	-	-	$\surd$
G_{10}	$0 \Rightarrow 0$	No	-	-	$\surd$

a result, the procedure continues with justification of $G_8 = 0$, which yields numerous other implications, as shown in Table II.

Identifying only direct implications seems to be a minor problem by itself, as demonstrated through the example in Fig. 2. However, the missed indirect implications often preclude many other implications from being found, as illustrated through the example in Fig. 3. Experiments from [27] indicate that this is a frequent phenomenon in logic circuits, and many possible implications will be missed if indirect implications are not identified.

V. CONDITIONS FOR ADDING REDUNDANT WIRES

In the previous section, we illustrated how to identify candidate redundant wires for a specific logic implication in the circuit. Thus, SET on the implication path can be masked by connecting the source of the implication as an additional input to the gates driven by the target of the implication. The addition of wires, however, should preserve the functionality of the circuit, i.e. the wires should be functionally redundant. An implication from source W_s to target W_t does not mean that W_s is equivalent to W_t. Therefore, in order to connect a wire originating from W_s to a gate driven by W_t, the following **two conditions** should hold true.

Condition 1: A wire can only be added to a logic gate if the output of the gate is correct regardless of the value of the wire, i.e. the value implicated on W_t is a *controlling value* of the gate. For example, consider the implication $(h, 0) \Rightarrow (G_8, 0)$ in the circuit in Fig. 1. Based on this implication, a wire originating from h can only be connected to AND (NAND) gates for which, if $G_8 = 0$, the output of the gate is 0 (1), regardless of the value of h.

TABLE III
CONDITIONS FOR ADDING REDUNDANT WIRES

Gates	Logic Implications			
	$0 \Rightarrow 0$	$0 \Rightarrow 1$	$1 \Rightarrow 0$	$1 \Rightarrow 1$
AND	$\Rightarrow$	$-$	$invert \Rightarrow$	$-$
NAND	$\Rightarrow$	$-$	$invert \Rightarrow$	$-$
OR	$-$	$invert \Rightarrow$	$-$	$\Rightarrow$
NOR	$-$	$invert \Rightarrow$	$-$	$\Rightarrow$
NOT	$\Rightarrow$ NAND	$-$	$-$	$\Rightarrow$ NOR

Condition 2: The polarity of the redundant wire must be the same as the polarity of the implicated value. For example, based on the implication $(e, 1) \Rightarrow (G_8, 0)$ in Fig. 1, a functionally redundant wire originating from e must be inverted before being connected to logic gates driven by G_8.

These two conditions are illustrated in Table III for all basic gates, and logic implications. A row in the table corresponds to a gate where W_t is an input, a column corresponds to a logic implication of W_s on W_t, and an entry indicates whether W_s can be connected, possibly inverted, to the gate driven by W_t or not. For example, if W_t is connected to an OR gate, and $(W_s, 1) \Rightarrow (W_t, 1)$, then W_s can be connected to the OR gate. However, if $(W_s, 0) \Rightarrow (W_t, 1)$, then W_s has to be first inverted, and then added to the OR gate. Finally, if W_t is connected to an inverter, and $(W_s, u) \Rightarrow (W_t, u)$, then the inverter is replaced by either a NAND gate, if $u = 0$, or a NOR gate, if $u = 1$. In this case, W_s and W_t are the inputs to the newly added gate, and the output of the gate is connected to the fanout gates of the replaced inverter.

In addition to these two conditions, we need to ensure that the circuit remains combinational after the addition of a wire. For this reason, the source of the implication must not be in the transitive fanout of the target of the implication. Wires in the transitive fanout of a gate can be simply identified by performing reachability analysis from the target of the implication, and excluding all reachable wires from the set of potential sources. For example, G_{10} in Fig. 1 should be excluded as a source of an implication for G_8 because the latter, which is the target wire of the implication, reaches G_{10}. Therefore, the implication $(G_{10}, 0) \Rightarrow (G_8, 0)$ cannot be used.

VI. SELECTION ALGORITHM

The objective of the selection algorithm is to add to the circuit the minimal set of functionally redundant wires that maximizes the reduction in the SET sensitization probability. Selection of the optimal set of such wires, however, is NP-complete, and thus, computationally expensive. Therefore, we use a greedy heuristic. And at each step we select, and add to the circuit the wire that provides the maximum reduction in SET sensitization probability. The process is repeated until no additional functionally redundant wire reduces the SET sensitization probability further. Besides the greedy selection, our method employs heuristics to address two additional sources of complexity. The first one is the problem of accurately assessing the sensitization probability of SET in a circuit. To avoid exhaustive simulation of all input combinations for all SET, which is required for calculating the sensitization probability, we use *sampling*. More specifically, we evaluate the sensitization probability of each SET in the circuit by simulating a fixed number, M, of random input patterns. The second one is the problem of performing the above calculation for all candidate redundant wires in the circuit. To reduce the search space, we only consider the addition of functionally redundant wires which mask SET that reach a primary output with high probability. More specifically, while performing simulation to calculate the sensitization probability of SET, we keep track of the number of times that each SET propagates to an output, we rank them in descending order, and we select the top N elements in this ranked list of SET. For each of them, we then identify all the functionally redundant wires that can mask it using one of the methods discussed in Section IV. Finally, we evaluate the sensitization probability of the circuit with each of these candidate wires added, and we select the one that provides the maximum reduction in SET sensitization probability. Our method is outlined in Algorithm 1.

Algorithm 1: Redundant Wire Selection

```
repeat
    —Evaluate $P_{sens}(old)$ of the circuit via simulation of
    $M$ random patterns;
    —$P_{sens}(best) = P_{sens}(old)$;
    —Identify the $N$ SET that are most frequently
    propagated to the output of the circuit;
    for $i = 1 \ldots N$ do
        —Identify the $k_i$ redundant wires that mask the
        $i$th SET;
        for $j = 1 \ldots k_i$ do
            —Add $j$-th wire to the circuit based on the
            rules in Table III;
            —Evaluate $P_{sens}(new)$ of the circuit;
            If $(P_{sens}(new) < P_{sens}(old))$
                —Best redundant wire $W_{best} = j$-th
                wire;
                —$P_{sens}(best) = P_{sens}(new)$;
        end
    end
    If $(P_{sens}(best) < P_{sens}(old))$
        —Add $W_{best}$ to the circuit;
until $P_{sens}(best) = P_{sens}(old)$;
```

While the above two heuristics significantly reduce the computation time of the proposed soft error mitigation method, large circuits imply that logic/fault simulation will require higher computation time in comparison to smaller ones. Moreover, the identification of redundant wires that realize logic implications will also require higher computation time. This is attributed to the complexity of the basic steps utilized by the proposed method (i.e. the time it takes to perform logic/fault simulation of a single input pattern, and to identify logic implications in the circuit), and not the method itself.

The addition of a functionally redundant wire introduces a new location where potential SET may occur. During the progress of the algorithm, and while functionally redundant wires are added, the overall SET sensitization probability may be reduced, while the total number of potential SET will always increase. Therefore, the benefit from adding functionally redundant wires reaches a point of diminishing returns, after which the addition of redundant wires would not only increase the total number of potential SET in the circuit, but also the overall SET sensitization probability. At this point, further addition of redundant wires would only increase the overall SER, and therefore the algorithm terminates.

VII. Experimental Results

In this section, we assess experimentally the SER reduction; and the hardware, power, and delay overhead of the proposed soft error mitigation method.

A. Setup

Algorithm 1 is applied on ISCAS'89 benchmarks [30], with $M = 250$, and $N = 25$.[3] This means that, in every step, fault simulation [31] of $M = 250$ random input patterns is used to identify the $N = 25$ most susceptible locations in the circuit, i.e. the locations where an SET has a high sensitization probability, P_{sens}, of reaching an output. The functionally redundant wire that yields the highest P_{sens} reduction is then added to the circuit, and the process is repeated until no additional reduction in P_{sens} is possible. Candidate functionally redundant wires are found by one of the three methods described in Section IV: backward justification [22], the direct implication identification procedure of FAN [23], or Recursive Learning [27]. The SER of the original, and the final circuit are evaluated using SERA [6], a soft error rate analysis tool. The hardware overhead is computed based on transistor counts of the original, and the final circuit. The power overhead is computed through the internal BDD power simulator of SIS [32]. The circuit is then mapped to the standard *lib2.genlib* library, and the delay overhead is computed based on the delay of the most critical path. The experiments were performed on a Sun Workstation with a 440MHz UltraSPARC IIi CPU, and 512MB RAM.

The results are reported in Tables IV–VI, for each of the three implication identification methods, respectively. We provide circuit details in the first five columns: name, number of inputs, number of outputs, gate count, and inverter count. In the next three columns, we report the number of implications identified in the circuit, the number of redundant wires that are added, and the CPU seconds required. In the following two columns, we indicate the percentile reduction in P_{sens}, and the SER of the circuit. In the last three columns, we report the percentile overhead in terms of hardware, power, and delay.

[3]Our experimental results indicate that increasing the value of N beyond 25 would yield marginal reduction to the probability of sensitization at a significant increase in computation time; thus, increasing the value of N beyond 25 is not beneficial.

TABLE IV
EXPERIMENTAL RESULTS ON ISCAS'89 BENCHMARK CIRCUITS USING BACKWARD JUSTIFICATION

Name	PI	PO	Gates	Inv.	Implications			Reduction (%)		Overhead (%)		
					Total	Added	Time	$\mathbf{P}_{sens}$	**SER**	Hardware	Power	Delay
s298	17	20	75	44	460	4	0.03	4.08%	3.07%	1.64%	1.11%	5.45%
s344	24	26	101	59	722	3	0.04	1.51%	2.34%	1.49%	5.97%	2.78%
s641	54	43	107	272	5348	7	0.29	5.83%	5.70%	2.78%	14.14%	10.17%
s386	13	13	118	41	892	10	0.04	5.49%	5.53%	2.88%	5.51%	12.30%
s444	24	27	119	62	826	18	0.05	10.16%	7.92%	9.66%	19.56%	14.23%
s713	54	42	139	254	4304	20	0.36	7.04%	7.91%	5.41%	31.00%	13.92%
s526	24	27	141	52	854	6	0.05	3.51%	2.79%	2.25%	3.53%	5.23%
s510	25	13	179	32	1127	27	0.07	21.60%	17.59%	18.63%	39.02%	38.12%
s820	23	24	256	33	1338	36	0.08	12.03%	14.37%	7.66%	9.92%	0.00%
s832	23	24	262	25	1321	30	0.08	11.09%	9.76%	6.11%	8.39%	0.83%
s953	45	52	311	84	3787	45	0.17	27.81%	32.26%	13.46%	17.77%	71.13%
s1196	32	32	388	141	2960	31	0.27	15.32%	11.01%	8.92%	16.63%	21.78%
s1238	32	32	428	80	2746	33	0.23	15.25%	13.97%	10.09%	16.76%	36.16%
s1423	91	79	490	167	2874	56	0.37	12.04%	10.70%	11.86%	27.91%	21.96%
s1488	14	25	550	103	2886	30	0.36	5.59%	2.43%	2.26%	14.31%	3.10%
s1494	14	25	558	89	2829	25	0.35	5.24%	3.18%	2.15%	12.98%	5.58%
					Average			10.22%	9.41%	6.71%	15.29%	16.43%

TABLE V
EXPERIMENTAL RESULTS ON ISCAS'89 BENCHMARK CIRCUITS USING DIRECT IMPLICATIONS

Name	PI	PO	Gates	Inv.	Implications			Reduction (%)		Overhead (%)		
					Total	Added	Time	$\mathbf{P}_{sens}$	**SER**	Hardware	Power	Delay
s298	17	20	75	44	1557	4	0.13	4.08%	3.07%	1.64%	1.11%	5.45%
s344	24	26	101	59	2185	4	0.21	2.19%	2.83%	2.23%	6.07%	0.00%
s641	54	43	107	272	7857	8	1.72	6.11%	5.94%	2.78%	14.14%	10.17%
s386	13	13	118	41	3714	10	0.22	5.49%	5.53%	2.88%	5.51%	12.30%
s444	24	27	119	62	1807	19	0.26	11.34%	8.03%	10.23%	19.44%	14.16%
s713	54	42	139	254	6749	20	1.67	7.04%	7.91%	5.41%	31.00%	13.92%
s526	24	27	141	52	2860	6	0.32	3.56%	2.45%	2.02%	2.06%	4.99%
s510	25	13	179	32	8771	29	0.46	23.76%	16.47%	17.92%	40.34%	38.19%
s820	23	24	256	33	17504	27	0.79	17.03%	16.43%	5.28%	6.15%	0.00%
s832	23	24	262	25	17265	28	0.78	16.87%	15.64%	5.85%	5.85%	0.00%
s953	45	52	311	84	28949	43	1.84	40.76%	36.84%	16.82%	21.48%	31.43%
s1196	32	32	388	141	24739	39	2.93	20.70%	27.61%	12.49%	22.47%	28.85%
s1238	32	32	428	80	26137	55	2.77	34.81%	34.76%	18.16%	30.26%	40.77%
s1423	91	79	490	167	12237	77	3.08	14.95%	17.09%	16.07%	29.01%	36.62%
s1488	14	25	550	103	52434	30	3.59	5.59%	2.43%	2.26%	14.31%	3.10%
s1494	14	25	558	89	50107	60	3.58	16.60%	11.51%	14.79%	15.94%	19.27%
					Average			14.43%	13.41%	8.55%	16.57%	16.20%

B. Results of Backward Justification

The results when using backward justification to identify implications are summarized in Table IV. The average reduction in SET sensitization probability is 10.22%, while the maximum reduction is 27.81% for benchmark circuit $s953$. The average SER reduction reported by SERA is 9.41%, and the maximum reduction is 32.26%. As can be observed, the reduction in P_{sens} is strongly correlated with the reduction in the SER of the circuits. The incurred hardware overhead is typically smaller than the achieved SER reduction. On average, hardware increases by 6.71%, and in the worst case, the hardware overhead is 18.63% for $s510$. Even with this increase in hardware, which introduces additional possible locations where a SET may occur, the SER is reduced by 17.59%. Power consumption increases, on average, by 15.29%; and delay increases, on average, by 16.43%. For several benchmark circuits, however, the SER reduction is significantly higher than the corresponding power & delay overhead. For example, the power consumption of $s953$ increases by 17.77%, while the SER is reduced by 32.26%. Similarly, the SER of $s820$ is reduced by 14.37%, while no delay overhead is incurred.

C. Results of Direct Implications

The results when using direct implications are summarized in Table V. The direct implication identification procedure identifies 7.42× more implications than backward justification, and requires 7.86× more CPU time. The number of implications added by Algorithm 1 increases, on average, by 18.40%. In some circuits, such as $s820$, $s832$, and $s953$, fewer functionally redundant wires are added to the circuit. Nevertheless, the reduction in P_{sens} is higher than the reduction in P_{sens} reported in Table IV. In these circuits, we clearly observe the benefits of direct implications, as fewer implications provide a higher reduction in P_{sens}. The average P_{sens} reduction across all circuits is 14.43%, while the maximum reduction is 40.76% for circuit $s953$. The average SER reduction is 13.41%, while the maximum reduction is 36.84% for circuit $s953$. Once again, the reduction in P_{sens}, and in SER are strongly correlated. The average hardware, power, and delay overhead is 8.55%, 16.57%, and 16.20%, respectively. In short, for an additional 1.84% hardware overhead, and 1.28% power overhead, direct implications provide an additional reduction of 4.21% to P_{sens}, 4.00% to SER, and 0.23% to the delay.

TABLE VI
EXPERIMENTAL RESULTS ON ISCAS'89 BENCHMARK CIRCUITS USING INDIRECT IMPLICATIONS

Name	PI	PO	Gates	Inv.	Implications			Reduction (%)		Overhead (%)		
					Total	Added	Time	$\mathbf{P}_{sens}$	**SER**	Hardware	Power	Delay
s298	17	20	75	44	1971	4	403	5.32%	6.86%	2.05%	1.82%	0.00%
s344	24	26	101	59	2378	4	427	2.19%	2.83%	2.23%	6.07%	0.00%
s641	54	43	107	272	8971	8	931	6.11%	5.94%	2.78%	14.14%	10.17%
s386	13	13	118	41	3762	10	137	5.49%	5.53%	2.88%	5.51%	12.30%
s444	24	27	119	62	2450	21	933	12.95%	7.94%	10.80%	20.65%	14.14%
s713	54	42	139	254	9007	16	1125	8.48%	12.49%	4.74%	28.90%	23.35%
s526	24	27	141	52	3203	9	4223	5.03%	8.70%	3.37%	4.11%	11.72%
s510	25	13	179	32	9085	29	4354	23.76%	16.47%	17.92%	40.34%	38.19%
s820	23	24	256	33	24763	36	67412	17.36%	17.02%	7.27%	7.60%	0.00%
s832	23	24	262	25	27856	35	88849	17.14%	19.74%	7.15%	6.70%	0.00%
s953	45	52	311	84	32015	45	21020	41.97%	40.45%	17.49%	21.74%	29.70%
s1196	32	32	388	141	27445	46	234116	23.73%	26.30%	15.07%	29.17%	30.73%
s1238	32	32	428	80	27127	60	319916	37.98%	35.78%	21.35%	36.76%	37.67%
s1423	91	79	490	167	12654	77	42380	14.95%	17.09%	16.07%	29.01%	36.62%
s1488	14	25	550	103	54922	28	19804	9.80%	7.21%	6.99%	17.63%	12.48%
s1494	14	25	558	89	58197	65	34507	18.47%	14.62%	17.49%	23.87%	29.83%
						Average		15.67%	15.31%	8.64%	18.38%	17.94%

D. Results of Indirect Implications

The results when using indirect implications are summarized in Table VI. Recursive Learning was applied on the benchmark circuits with a recursion depth of two, which is sufficient to identify the majority of implications in a circuit [27]. The Recursive Learning procedure requires around 30,000× more CPU time, and identifies 1.18× more implications than the direct implication identification procedure. The increase in CPU time is very substantial, and yields many more implications; yet the number of added implications to the circuit increases by only 8.26%, which translates to a moderate reduction of 1.24% in P_{sens}, and 2.10% in SER over direct implications. Furthermore, the hardware, power, and delay overhead reported in Table VI are comparable to those reported in Table V. In short, the significant increase in CPU time for identifying indirect implications, along with the small additional SER reduction achieved over direct implications, suggests that the latter provide the most attractive option.

E. Trade-off Exploration

The proposed selection algorithm aims at assessing the effectiveness of adding functionally redundant wires in reducing the SER of a circuit. As such, Algorithm 1 continues to select & add functionally redundant wires to the design, regardless of the incurred hardware overhead, as long as the P_{sens} continues to reduce. Yet it is possible that interim solutions may constitute a better trade-off point. For example, in Fig. 4, we show the interim results of adding redundant wires identified through backward justification for $s953$. The reduction in P_{sens}, normalized based on the maximum achievable P_{sens} reduction (27.81%), is plotted against the incurred hardware overhead, also normalized based on the total hardware overhead (13.46%). As seen on the figure, the hardware overhead curve is monotonic with the SER reduction. This is expected because the selection algorithm adds the wire that provides the maximum reduction in the SER of the circuit. However, the curve also shows that there are many interim circuits on which similar P_{sens} reduction to the final circuit is achieved, yet at a much lower cost. For example, 90% of the total possible P_{sens} reduction can be achieved by only 40%

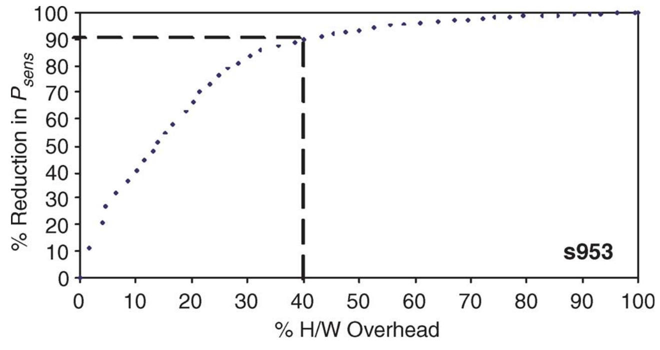

Fig. 4. Reduction in P_{sens} vs. H/W overhead.

of the corresponding hardware overhead. In other words, while the maximum P_{sens} reduction of 27.81% incurs an overhead of 13.46%, a slightly lower P_{sens} reduction of 25.03% incurs an overhead of only 5.38%. Therefore, if the resources are limited, the selection step can be terminated when the overhead, in terms of hardware in this example, is exceeded. To assess the potential of adding functionally redundant wires, we opted to use a selection algorithm that returns the logic implementation with the maximum achievable SER reduction. The proposed algorithm can be easily modified to drive the selection of functionally redundant wires based on a user-specified cost function including hardware, power, and/or delay overhead; and to terminate the selection procedure if a user-specified target on the probability of sensitization is reached.

VIII. CONCLUSIONS

Logic implications constitute a source of fine-grained invariance that can be exploited to mitigate soft errors. More specifically, the addition of functionally redundant wires that realize these logic implications reduces the susceptibility of logic circuits to SET. Identification of logic implications, and selective addition of functionally redundant wires through the methods and algorithms described herein, result in significant SER reduction, commensurate with the incurred hardware, power, and delay overhead. The proposed mitigation method averts soft errors at the gate level; and thus it is orthogonal to, and can be combined with, circuit-level SER reduction methods such as gate and transistor resizing, and/or hardened flip-flop designs.

REFERENCES

[1] P. Shivakumar, M. Kistler, S. W. Keckler, D. Burger, and L. Alvisi, "Modeling the effect of technology trends on the soft error rate of combinational logic," in *International Conference on Dependable Systems and Networks*, 2002, pp. 389–398.

[2] L. Lantz, "Tutorial: Soft errors induced by alpha particles," *IEEE Trans. Reliability*, vol. 45, no. 2, pp. 174–179, 1996.

[3] P. Lidén, P. Dahlgren, R. Johansson, and J. Karlsson, "On latching probability of particle induced transients in combinational networks," in *Symposium on Fault-Tolerant Computing*, 1994, pp. 340–349.

[4] K. Mohanram and N. A. Touba, "Cost-effective approach for reducing soft error failure rate in logic circuits," in *International Test Conference*, 2003, pp. 893–901.

[5] C. Zhao, X. Bai, and S. Dey, "A scalable soft spot analysis methodology for noise effects in nano-meter circuits," in *Design Automation Conference*, 2004, pp. 894–899.

[6] M. Zhang and N. R. Shanbhag, "A soft error rate analysis (SERA) methodology," in *International Conference on Computer-Aided Design*, 2004, pp. 111–118.

[7] S. Krishnaswamy, G. F. Viamonte, I. L. Markov, and J. P. Hayes, "Accurate reliability evaluation and enhancement via probabilistic transfer matrices," in *Design, Automation and Test in Europe Conference*, 2005, pp. 282–287.

[8] B. Zhang, W. S. Wang, and M. Orshansky, "FASER: Fast analysis of soft error susceptibility for cell-based designs," in *International Symposium on Quality Electronic Design*, 2006, pp. 755–760.

[9] N. Miskov-Zivanov and D. Marculescu, "Circuit reliability analysis using symbolic techniques," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 12, no. 25, pp. 2638–2649, 2006.

[10] J. M. Cazeaux, D. Rossi, M. Omana, C. Metra, and A. Chatterjee, "On transistor level gate sizing for increased robustness to transient faults," in *International On-Line Testing Symposium*, 2005, pp. 23–28.

[11] Y. S. Dhillon, A. U. Diril, and A. Chatterjee, "Soft-error tolerance analysis and optimization of nanometer circuits," in *Design, Automation and Test in Europe Conference*, 2005, pp. 288–293.

[12] N. Miskov-Zivanov and D. Marculescu, "MARS-C: modeling and reduction of soft errors in combinational circuits," in *Design Automation Conference*, 2006, pp. 767–772.

[13] Q. Zhou and K. Mohanram, "Gate sizing to radiation harden combinational logic," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 1, pp. 155–166, 2006.

[14] T. Monnier, F. M. Roche, J. Cosculluela, and R. Velazco, "SEU testing of a novel hardened register implemented using standard CMOS technology," *IEEE Trans. Nuclear Science*, vol. 46, no. 6, pp. 1440–1444, 1999.

[15] M. Omana, D. Rossi, and C. Metra, "Novel transient fault hardened static latch," in *International Test Conference*, 2003, pp. 886–892.

[16] M. Nicolaidis, "Time redundancy based soft-error tolerance to rescue nanometer technologies," in *VLSI Test Symposium*, 1999, pp. 86–94.

[17] P. Elakkumanan, K. Prasad, and R. Sridhar, "Low power SER tolerant design to mitigate single event transients in nanoscale circuits," *ASP Journal of Low Power Electronics*, vol. 1, pp. 182–193, 2005.

[18] V. Joshi, R. Rao, D. Sylvester, and D. Blaauw, "Logic SER reduction through flip-flop redesign," in *International Symposium on Quality Electronic Design*, 2006, pp. 611–616.

[19] D. G. Mavis and P. H. Eaton, "Soft error rate mitigation techniques for modern microcircuits," in *International Reliability Physics Symposium*, 2002, pp. 216–225.

[20] S. Mitra, N. Seifert, M. Zhang, Q. Shi, and K. S. Kim, "Robust system design with built-in soft-error resilience," *IEEE Computer*, vol. 38, no. 2, pp. 43–52, 2005.

[21] P. Elakkumanan, K. Prasad, and R. Sridhar, "Time redundancy based scan flip-flop reuse to reduce SER of combinational logic," in *International Symposium on Quality of Electronic Design*, 2006, pp. 617–624.

[22] M. L. Bushnell and V. D. Agrawal*, Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits*. : Kluwer Academic Publishers, 2000.

[23] H. Fujiwara and T. Shimono, "On the acceleration of test generation algorithms," *IEEE Trans. Computers*, vol. 32, no. 12, pp. 1137–1144, 1983.

[24] M. Schulz, E. Trischler, and T. Sarfert, "SOCRATES: A highly efficient automatic test pattern generation system," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 7, no. 1, pp. 126–137, 1988.

[25] M. Schulz and E. Auth, "Improved deterministic test pattern generation with applications to redundancy identification," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 8, no. 7, pp. 811–816, 1989.

[26] W. Kunz and D. K. Pradhan, "Accelerated dynamic learning for test generation in combinational circuits," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 12, no. 5, pp. 684–694, 1993.

[27] W. Kunz and D. K. Pradhan, "Recursive learning: A new implication technique for efficient solutions to CAD-problems: Test, verification and optimization," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 13, no. 9, pp. 1149–1158, 1994.

[28] J. Zhao, E. Rudnick, and J. Patel, "Static logic implication with application to fast redundancy identification," in *VLSI Test Symposium*, 1997, pp. 288–293.

[29] W. Kunz, D. Stoffel, and P. R. Menon, "Logic optimization and equivalence checking by implication analysis," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 16, no. 3, pp. 1149–1158, 1997.

[30] ISCAS'89 Benchmark Circuits Information [Online]. Available: http://www.cbl.ncsu.edu

[31] H. K. Lee and D. S. Ha, "HOPE: An efficient parallel fault simulator for synchronous sequential circuits," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, no. 9, pp. 1048–1058, 1996.

[32] E. M. Sentovich, K. J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldahna, H. Savoj, P. R. Stephan, R. K. Brayton, and A. Sangiovanni-Vincentelli, "SIS: A System for Sequential Circuit Synthesis," EECS UC Berkeley, CA 94720, ERL MEMO. No. UCB/ERL M92/41, 1992.

Sobeeh Almukhaizim (S'99–M'07) received the Diploma of Electrical and Computer Engineering from Kuwait University, Kuwait, in 1999; the M.S. in Computer Science and Engineering from the University of California, San Diego, in 2001; and the M.S., M.Phil, and Ph.D. in Electrical Engineering from Yale University in 2003, and 2007. He is currently an Assistant Professor of Computer Engineering at Kuwait University. His research interests include VLSI design and test, computer architecture, microprocessor testing, fault tolerance, test and reliability of synchronous and asynchronous circuits, and soft error mitigation in digital circuits.

Yiorgos Makris (S'96–A'01–M'03) received the Diploma of computer engineering and informatics from the University of Patras, Greece, in 1995; and the M.S., and Ph.D. degrees in computer science and engineering from the University of California, San Diego, in 1997, and 2001, respectively. He is currently an Associate Professor of Electrical Engineering and Computer Science at Yale University, where he leads the Testable and RELiable Architectures (TRELA) research group. His current research interests include soft-error mitigation in digital circuits, machine learning-based testing of analog/RF circuits, as well as test and reliability of asynchronous circuits.