X-Masking During Logic BIST and Its Impact on Defect Coverage

Yuyi Tang, Hans-Joachim Wunderlich, *Member, IEEE*, Piet Engelke, *Student Member, IEEE*, Ilia Polian, *Member, IEEE*, Bernd Becker, *Senior Member, IEEE*, Jürgen Schlöffel, *Member, IEEE*, Friedrich Hapke, and Michael Wittke

***Abstract*—We present a technique for making a circuit ready for logic built-in self test by masking unknown values at its outputs. In order to keep the silicon area cost low, some known bits in output responses are also allowed to be masked. These bits are selected based on a stuck-at n-detection based metric, such that the impact of masking on the defect coverage is minimal. An analysis based on a probabilistic model for resistive short defects indicates that the coverage loss for unmodeled defects is negligible for relatively low values of n.**

***Index Terms*—Defect coverage, logic built-in self test (BIST), resistive bridging faults (RBFs), X-masking.**

I. Introduction

BUILT-IN self test (BIST) solves many of today's testing problems, including pin throughput issues, complexity of test programs and test application at speed, and enables in-field testing [1]. While BIST became industry standard for memories in the 1990s [2], there are still some obstacles for its application to random logic. One class of circuits that are difficult to handle using logic BIST (LBIST) consists of those that produce unknown values (X values) at the outputs. Sources of unknown values include tri-stated or floating buses, uninitialized flip-flops or latches, signals that cross clock domains in circuits with multiple clock domains, and X values coming from analog or memory blocks that are embedded in the random logic circuit. If an unknown value is fed into a test response evaluator (TRE), the signature can be affected. For the most popular TRE, the multiple input signature register (MISR), a single X value invalidates the whole signature.

This problem has been attacked from two directions. First, *X-tolerant compactors*, i.e., TREs that are less vulnerable to X values, have been proposed, including X-COMPACT by Intel [3] and Convolutional Compactor by Mentor Graphics [4]. The second solution puts no restriction on the type of TRE used. The unknown values that appear at the outputs of the circuit are *masked out* by additional logic, such that only known values are fed into the TRE [5]–[8]. X-tolerant compactors are space compactors. They are typically designed such that they can tolerate a certain number of Xs in addition to a number of faulty bits.[1] While their area overhead is larger than for space compactors without X tolerance, the exact overhead is a function of the assumed maximal number of X values which can be present at the same time. In contrast, masking is test set specific. It can be used with space or time compaction. Its overhead depends on implementation, e.g., whether any information is stored in the tester [7]. It can be employed in a scheme that protects intellectual property (IP). The technique proposed here is of the second type, although it tackles problems which also exist for X-tolerant compactors, as will be explained below. The *X-masking logic* (XML) is introduced between the circuit under test (CUT) and the TRE. It consists of OR gates and synthesized control logic. The first input of each OR gate is connected to an output of the CUT, while the second input originates from the control logic. When the control logic produces a logic-1, the output of the OR gate is forced to logic-1, and hence the response of the CUT is masked. The control logic is a combinational function that uses as inputs the pattern counter and bit counter, which are generally part of the LBIST test control logic for controlling the number of applied patterns and the scan shift/capture cycles.

In principle, it is possible to mask out only the unknown values in the response and to leave unchanged all the other values. However, masking the unknown bits exactly would result in high silicon area cost of XML. Furthermore, this is not necessary, as the vast majority of faults are detected by many different patterns. Fig. 1 shows the number of detections per stuck-at fault for the ISCAS circuit s5378, which is also representative for other circuits. It indicates that not all known bits are actually required for detection. Hence, we allow also some of the known bits to be masked out, in a way that the stuck-at fault coverage is not compromised (earlier works also used this approach [5], [6]). However, the coverage of *unmodeled defects* might be affected by masking out known bits. To reduce the likelihood of coverage loss for unmodeled defects, we introduce more conservative requirements for allowing a known bit to be masked out.

[1]They may be effective in presence of a higher number of Xs or faulty bits with a certain probability. See [9] for a detailed study of such probabilities in case of convolutional compactors.

Manuscript received June 14, 2005; revised November 7, 2005. This work was supported in part by the German Federal Ministry of Education and Research (BMBF) in the Project AZTEKE under Contract 01M3063C. This paper has been published in part in the *Proceedings of International Test Conference*, 2004.

Y. Tang and H.-J. Wunderlich are with the Institute of Computer Architecture and Computer Engineering, University of Stuttgart, D-70569 Stuttgart, Germany (e-mail: Yuyi.Tang@informatik.uni-stuttgart.de; wu@informatik.uni-stuttgart.de).

P. Engelke, I. Polian, and B. Becker are with the Institute for Computer Science, Albert-Ludwigs-University, D-79110 Freiburg i. Br., Germany (e-mail: engelke@informatik.uni-freiburg.de; polian@informatik.uni-freiburg.de; becker@informatik.uni-freiburg.de).

J. Schlöffel, F. Hapke, and M. Wittke are with the Philips Semiconductors GmbH Design Technology Center, D-21147 Hamburg, Germany (e-mail: juergen.schloeffel@philips.com; friedrich.hapke@philips.com; michael.wittke@philips.com).

Digital Object Identifier 10.1109/TVLSI.2005.863742

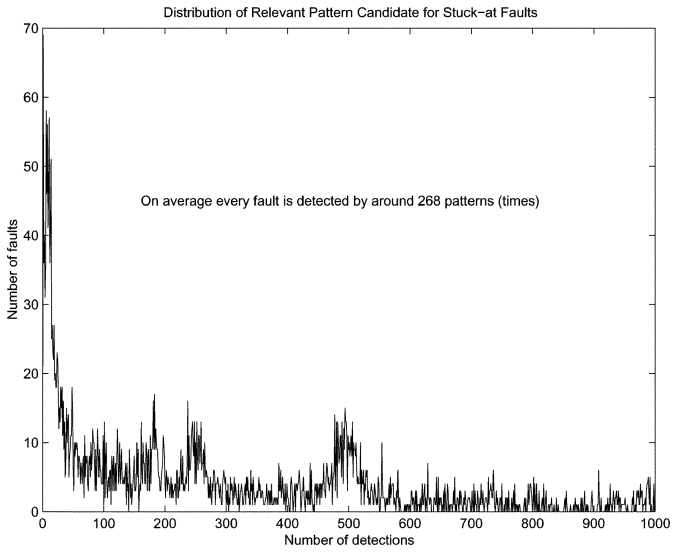

Fig. 1. Number of detections for stuck-at faults of s5378 (1000 random patterns).

The requirements are based on n-detection [10], [11] that has been demonstrated to lead to test sets with high defect coverage (a recent study is reported in [12]). In general, introducing XML will lower the number of times a stuck-at fault is detected (even if each fault is still detected at least once). For a given parameter $n \geq 1$, the number of detections for a stuck-at fault must not decline below n due to masking. For instance, assume that a stuck-at fault is detected five times without masking of known bits, and let n be 3. Then, it is acceptable that the number of detections with XML drops to 4 or 3, but not below. Increasing n leads to a higher number of stuck-at fault detections (and hence hopefully to a better coverage of unmodeled defects) but also to larger silicon area for the XML.

In this article, we study the impact of masking on unmodeled defects for the proposed architecture. For this purpose, we consider resistive bridging faults (RBF) [13], [14] as surrogates of unmodeled defects. The RBF model [15]–[17] takes into account several nontrivial electrical properties of resistive defects, such as pattern dependency. Using the simulator from [17], we compute the RBF coverage with and without masking of known bits. Note that the information on RBF coverage is not available to the XML synthesis procedure, which is guided by stuck-at detection information only. For different values of n we obtain different implementations of XML which trade off unmodeled defect coverage vs. silicon area cost. It turns out that the difference in RBF coverage with and without XML is not significant, and for $n \geq 5$ it practically disappears.

Current advanced X-masking solutions employ **<DEFINE LFSR.>** LFSRs in combination with reseeding [6], weight assignment to individual stages of the LFSR [8] or reseeding with tester support [7]. For a given set of responses, an LFSR generates control signals for masking. Similar to our method, the technique from [6] accepts masking of some of the known bits as long as the stuck-at fault coverage is not sacrificed. The LFSR seeds are stored on-chip. However, the issue of unmodeled defects is not dealt with in the mentioned papers. In contrast, we use n-detection information and study, for the first time, the trade-off between unmodeled defect coverage and the size of the logic. It turns out that the proposed XML requires less area than the LFSR-based architecture from [6], although we use a higher probability of X appearance.

Although our article focuses on X masking, the potential decrease of unmodeled defect coverage is also an issue for X-tolerant compactors [3], [4]. They are based on connecting a circuit output to multiple XOR trees. As a consequence, unknown values on outputs may invalidate detections on circuit outputs connected to the same XOR gates. Providing more compactor outputs (XOR trees) will increase the circuit area and reduce the probability that a defect is missed. Hence, the trade-off between area cost and unmodeled defect coverage, which is under investigation in this article for the case of X masking, exists also for X-tolerant compactors.

The remainder of the article is structured as follows: In Section II, the XML is introduced and its synthesis is explained. Essential information on the RBF model is summarized in Section III. The experimental setup is described and the results are reported in Section IV. Section V concludes the article.

II. X-Masking Logic

A. Problem Formulation

Let the CUT have p outputs, and let the test set consist of q patterns. Let the responses of the CUT be $(r_{11}, r_{12}, \ldots r_{1p})$, $(r_{21}, r_{22}, \ldots r_{2p})$, $(r_{q1}, r_{q2}, \ldots r_{qp})$, where $r_{ij} \in \{0, 1, X\}$ is the value that appears at the jth output of the CUT as a response to the ith test pattern in absence of any fault. The term "output" stands for "primary output" for combinational and nonscan sequential circuits, scanout ports for full-scan circuits and primary outputs and scan-outs for partial-scan circuits. We are looking for a function $\mathrm{XML} : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{B}$ such that $\mathrm{XML}(i, j) = 1$ if $r_{ij} = X$ (i.e., all unknown values are masked). Furthermore, some r_{ij} that are important for preserving the fault coverage (called *relevant bits*) must *not* be masked ($\mathrm{XML}(i, j) = 0$ must hold for these bits). In general, there are several possibilities to select the set of relevant bits such that the desired fault coverage can be achieved. The size of X-masking logic depends on the number and exact positions of relevant bits. The algorithm for selection of relevant bits in a way that leads to compact XML blocks will be explained in Section II-C. For values of (i, j), for which $r_{ij} \neq X$ and which are not among the relevant bits, XML is allowed to assume either 0 or 1. This degree of freedom is utilized for minimizing the XML logic, as introduced next.

B. Implementation

We describe the implementation of XML for deterministic LBIST (DLBIST) based on bit flipping [18], [19]. However, the technique does not impose any constraints on the used pattern generator and TRE. Thus, it can be adapted to other LBIST architectures or test compression.

Fig. 2 shows the DLBIST architecture without XML. An LFSR is used as the source of random patterns. In order to achieve the desired fault coverage, some of the bits produced by the LFSR are inverted, which is controlled by *bit-flipping logic* (BFL) (referred to in [20] as bit-fixing logic). BFL is a combinational block that takes the LFSR state, the pattern

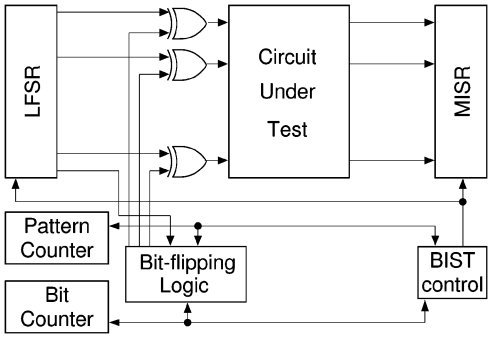

Fig. 2. DLBIST without XML.

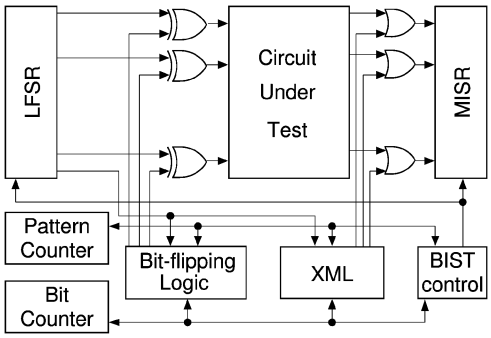

Fig. 3. DLBIST with XML.

number (from the pattern counter) and the bit number (from the bit counter) and selects the LFSR outputs to be inverted by driving a logic-1 at the inputs of the corresponding XOR gates. The responses of the CUT are fed into a MISR.

The DLBIST architecture with XML is shown in Fig. 3. Similarly to BFL, XML is a combinational logic block that has the LFSR state, the pattern number and the bit number as inputs. XML provides control signals to the OR gates between the CUT and the MISR. A bit is masked iff XML generates a logic-1 at the corresponding OR gate. Note that XML is not on the critical path of the CUT. The impact on the circuit delay is due to the added OR gates only, as long as the delay of XML does not exceed the delay of CUT itself.

The problem to synthesize the XML can be formulated as an instance of logic synthesis with don't care (DC) sets.[21]. The value at the jth output of the CUT when the ith test pattern is applied is uniquely determined by the triple (LFSR state, pattern number, bit number), i.e., a state of (LFSR, pattern counter, bit counter). With the notation of Section II-A, the logic synthesis instance is composed as follows: the ON set consists of (LFSR, pattern counter, bit counter) state triples that correspond to (i, j) with $r_{ij} = X$. The OFF set includes all those triples that correspond to *relevant bits* (the description of how the relevant bits are selected follows in Section II-C). All other triples constitute the DC set.

Once the ON and OFF sets are known, logic synthesis can be run. In general, compact ON and OFF sets will lead to smaller logic, because a logic synthesis tool has more degrees of freedom. While the ON set is given by the X values in the responses, there are several alternative OFF sets, depending on which bits are selected as relevant. Thus, both the number of

Procedure select_rel_bits
Input: Fault isolation table FIT; parameter n
Output: Compact set RB of relevant bits
that fulfills coverage requirements
(1) $RB := \emptyset$;
(2) **while** (FIT not empty) **begin**
(3) $f :=$ fault from FIT with lowest number of detections;
(4) $RB := RB \cup$
select_bits_for_fault$(f, \min\{N_f, n\} - D_f)$;
// Select bits to ensure sufficient detections
(5) **for each** fault g from FIT **begin**
(6) Determine D_g with relevant bits selected so far;
(7) **if** $(D_g \geq \min\{N_g, n\})$
(8) **then** exclude g from FIT;
(9) **end for**
(10) **end while**
(11) **return** RB;
end select_rel_bits;

Fig. 4. Algorithm for selecting relevant bits.

relevant bits and the number of patterns they belong to should be minimized.

C. Selection of Relevant Bits

For the sake of simplicity, we call a value at an output j of the circuit when a test pattern i is applied a bit (so for p outputs and q patterns there are pq bits). A subset of these pq bits has to be selected as relevant bits that are excluded from masking. Remember that a triple (LFSR state, pattern number, bit number) corresponds to a bit. The triples corresponding to relevant bits are included into the OFF set of the logic synthesis problem formulated above. If more bits are selected as relevant, the number of fault detections, but also the silicon area cost is growing. As an additional constraint, there is a parameter n which is defined as the minimal number of detections that must be preserved when known bits are masked out. Obviously, a higher value of n requires more bits to be selected as relevant.

The selection algorithm uses the *fault isolation table* to select relevant bits. The fault isolation table contains for each stuck-at fault f all bits for which it is detected when no XML logic is introduced (the number of such bits is denoted as N_f). A bit is said to *detect* a fault if the fault's effect is observed at the output of the circuit for the test pattern that corresponds to the bit. For each fault f, the number of detections D_f must be guaranteed to be at least $\min\{N_f, n\}$. Note that if n bits detecting a fault have been selected as relevant, the actual number of detections will typically be higher, because the XML could (but is not guaranteed to) leave other bits detecting this fault (but not selected as relevant) unmasked.

The algorithm select_rel_bits is shown in Fig. 4. It constructs the set *RB* of relevant bits such that each fault f is detected by at least $\min\{N_f, n\}$ bits from *RB*. This is done iteratively. In each iteration, (Lines 2–10), a fault is picked and several bits are selected as relevant, such that the fault is detected by a sufficient number of bits ($D_f =$ number of detections of the fault f). The selected bits might also detect other faults. This is checked in Line 6. All faults g whose number of detections D_g is greater or equal than the required number $\min\{N_g, n\}$ are excluded from the fault isolation table (Line 7–8). Note that the fault f from Line 3 is always among these faults. The algorithm stops when the fault isolation table is empty (Line 2).

Procedure select_bits_for_fault
Input: Fault f, number M of bits to select
Output: M bits $b_1, b_2, \ldots, b_M$
(1) set_of_bits $SB := \emptyset$;
(2) **while** ($|SB| < M$) **begin**
(3) Select a pattern P with at least 1 bit detecting f (according to cost function $CF1$ – see Eq. (1));
(4) $SB := SB \cup$ bits of P that detect f;
(5) **end while** // Now, SB may contain more than M bits
(6) Sort SB according to cost function $CF2$ (see Eq. (2));
(7) **return** First M elements of SB;
end select_bits_for_fault;

Fig. 5. Procedure for selecting relevant bits for a single fault (bit-based).

The sub-routine select_bits_for_fault (which is called in Line 4 of Procedure select_rel_bits) has to select $M := \min\{N_f, n\} - D_f$ relevant bits that detect the fault f (where D_f is the number of detections of f by bits selected for other faults treated before f). The pseudo-code of Procedure select_bits_for_fault is shown in Fig. 5. The goal is to select bits from as few different patterns as possible. First, a suitable pattern is selected according to cost function $CF1$. Every pattern P is assigned a flag $\mathrm{New}(P)$, with $\mathrm{New}(P) = 0$ if any bit from P has already been selected as relevant and $\mathrm{New}(P) = 1$ otherwise. Let P be the kth pattern. Then

$$CF1(P) = \mathrm{New}(P) + \frac{1}{(|\{f|P \text{ detects } f\}| + 1)} + \frac{1}{(|\{r_{kj}|r_{kj} \neq X, 1 \leq j \leq p\}| + 1)}. \quad (1)$$

$CF1$ assigns lower cost to patterns already taken for some other faults and to patterns that detect a high number of faults. Also, patterns with a low number of unknown bits are preferred by $CF1$, because this helps to decouple unknown bits (ON set) and relevant bits (OFF set). Bits detecting f are collected (Lines 3 and 4). If there are less than M bits, then bits from an additional pattern are added (Line 2). At the end of the first stage, there is a pool of at least M bits (in at most M patterns), from which exactly M bits are selected according to the cost function $CF2$ (Line 6). $CF2$ of a bit r_{ij} (i.e., at the jth output of the ith pattern) is defined as

$$CF2(i,j) = |\{r_{ik}|r_{ik} = X, 1 \leq k \leq p\}| + |\{r_{kj}|r_{kj} = X, 1 \leq k \leq q\}|. \quad (2)$$

$CF2$ prefers a bit position that corresponds to circuit output j and pattern i such that the number of X values for pattern i and other circuit outputs and for output j and other patterns are minimal. (Again, this is done in order to decouple the ON-set from the OFF-set). The selected bits are added to *RB* in Line 4 of Procedure select_rel_bits in Fig. 4.

The computational complexity of Procedures select_rel_bits and select_bits_for_fault is analyzed next. We assume that the fault isolation table has been calculated as a pre-processing step and hence the complexity to decide whether a fault f is detected at the bit r_{ij} is $O(1)$. Cost function $CF2$ can also be calculated as a pre-processing step, so every call to $CF2$ has complexity $O(1)$. The value of cost function $CF1$ depends on the flag *New*, which is updated during the run time of the algorithm. Hence, only the second and the third term in (1) can be calculated in advance. The worst-case complexity of Line 3 of Procedure select_bits_for_fault is $O(q)$ as it could be necessary to check all the patterns. (Using known speed-up for priority queue implementation would not make the complexity logarithmic, as the searched pattern must satisfy two conditions: minimal $CF1$ and detection of fault f.) Since this operation is repeated up to M times, Lines (1–5) have a complexity of $O(M \cdot q)$. $CF1$ is updated for every selected pattern. After Line (5), *SB* has less than $M + p$ elements, M out of which (having minimal $CF2$) have to be selected. Using a heap representation, this can be done in $O(M \cdot \log(M + P))$. The overall complexity of Procedure select_bits_for_fault is $O(M \cdot (q + \log(M + p)))$.

The loop in Lines 2–10 of Procedure select_rel_bits is repeated up to F times, where F is the number of faults. Line 3 requires $O(\log F)$ if the FIT is represented by a heap. Overestimating $\min\{N_f, n\} - D_f$ by n, the complexity of Line 4 is $O(n \cdot (q + \log(n + p)))$. Lines 5–9 have a worst-case complexity of $O(F \cdot p \cdot q)$. The overall complexity of the method is $O(F \cdot (n \cdot (q + \log(n + p)) + F \cdot p \cdot q)$ plus the pre-processing time.

For comparison purposes, we implemented an alternative version of Procedure select_bits_for_fault. For a given n, it selects all bits from at least n patterns in which at least one bit detects the fault. If there are less than n such patterns then all the bits from all the patterns are selected. If the number of such patterns exceeds n, selection is made based on the cost function $CF1$ mentioned above. We refer to this relevant bits selection method as "pattern-based," while we call the method outlined above "bit-based." The pattern-based approach typically results in more bits selected as relevant than the bit-based method for the same value of n.

The proposed algorithms can treat mid-size circuits such as larger ISCAS benchmarks, because the complete fault isolation table is calculated in advance. Methods which compute required parts of the FIT on-the-fly may be required for industrial designs.

III. Resistive Bridging Fault Model (RBF)

In this section, we provide a brief overview of the RBF model, which is used as a surrogate of unmodeled defects in this article. The material here is restricted to concepts necessary for understanding the analysis in this article; see, e.g., [17] for an in-depth consideration.

The main difficulty when dealing with resistive faults is that, unlike for the nonresistive case, there is an unknown value to be taken into account, the resistance. This is because it cannot be known in advance which particle will cause the short defect corresponding to the bridge. Parameters like its shape, size, conductivity, exact location on the die, evaporation behavior and electromigration can influence the resistance of the short defect. A short defect may be detected by a test pattern for one resistance value, and the short between the same nodes may not be detected by the same pattern for another resistance. This fundamentally changes the meaning of standard testing concepts, like redundancy, fault coverage, and so forth.

In order to handle this ambiguity, Renovell *et al.* [15], [16] introduced the concept of *analogue detectability interval* (ADI) and probabilistic bridging fault coverage. The ADI of a fault is the range of bridge resistances for which the faulty logical value is produced at one or more circuit outputs. It is calculated based on electrical parameters of the logic gates at the bridge site. Most ADIs are intervals of the type $[0, R_{\rm crit}]$ (i.e., there is a resistance value $R_{\rm crit}$ such that all short defects between two nodes with resistance below this value are detected and all other defects between same two nodes are too weak to be detected), but the existence of different types of ADIs has been demonstrated in [16], [17].

The covered ADI (*C*-ADI) of a test set is defined as the union of the ADIs of individual test patterns. Lobal ADI (*G*-ADI) is the *C*-ADI of the exhaustive test set. Hence, *C*-ADI includes all the bridge resistances for which the fault has been detected by at least one test pattern, while *G*-ADI consists of all values of R_{sh} for which the fault is *detectable*. If *C*-ADI of a test set equals *G*-ADI, then this test set is as effective in detecting RBF as the exhaustive test set. A bridging fault with resistance not in *G*-ADI is redundant.

The RBF fault coverage (FC) [16], [17] is defined as

$$\mathrm{FC}(f) = 100\% \cdot \frac{\left(\int_{C\text{-ADI}} \rho(r)dr\right)}{\left(\int_{G\text{-ADI}} \rho(r)dr\right)}$$

where $\rho(r)$ is the probability density function of the short resistance r obtained from manufacturing data. Thus, FC relates *C*-ADI to *G*-ADI, weighted by the likelihood of different values of R_{sh}. In prior work, FC was referred to as *G*-FC to distinguish it from approximative metrics.

We will use resistive bridge fault coverage FC in the experiments of Section IV to estimate the impact of XML on the coverage of unmodeled defects.

IV. Experimental Results

We applied the XML synthesis approach to ISCAS 85 [22] and combinational parts of ISCAS 89 [23] circuits. Table I quotes the number of patterns in the test set (which are embedded into the LFSR sequence), the number of outputs of a circuit, its size in gate equivalents (GE) and the size of BFL. Note that no BFL is required if the pseudo-random sequence reaches 100% fault efficiency. As these circuits do not have tri-state buses or multiple clock domains, they do not produce X values at the outputs. Consequently, we assumed a scenario when a preceding block induces unknown values at the circuit's inputs. We used the test sets for stuck-at faults generated by a commercial tool and randomly injected X values at the inputs. Then, the X values have been propagated to the outputs using three-valued logic simulation and resulting in realistic correlations of unknown values at the outputs.

We performed two experiments: Experiment 1 and Experiment 2. In Experiment 1, X values were randomly injected at 1% of the inputs. In Experiment 2, 3% of input values (instead of 1%) were set to X. In order to obtain patterns with relatively large and relatively small fractions of unknown values, we distributed X values as follows: we defined a random variable y that assumes values between 0 and 6 (with uniform probability). For a pattern, we first assign a random value between 0 and 6 to y. Then, we set $y\%$ of the positions in the pattern to X (resulting, on average, in 3% unknown values across the patterns).

TABLE I
Information on the Employed Circuits

Circuit	Patterns	Outputs	Circuit size [GE]	BFL size [GE]
c0432	43	7	203	0
c0499	52	32	275	0
c0880	33	26	469	0
c1355	85	32	619	0
c1908	116	25	938	0
c2670	65	140	1566	0
c3540	126	22	1741	0
c5315	72	123	2608	0
c6288	21	32	2480	0
c7552	96	108	3827	433
cs00298	25	20	156	0
cs00344	16	26	210	0
cs00400	28	27	213	0
cs00444	28	27	232	0
cs00526	55	27	245	0
cs00713	31	42	489	0
cs05378	121	228	3221	0
cs13207	276	790	9441	367
cs15850	139	684	11067	686
cs38584	147	1730	22447	590

Logic synthesis has been performed using a tool based on BDD's (binary decision diagrams) developed at the University of Stuttgart in cooperation with Philips. Details on the logic synthesis procedure can be found in [24] (some of the features described in that paper were not available when the experiments were performed). For selecting relevant bits, we employed both the bit-based and the pattern-based approach (explained in Section II-C) with different values of n.

A. Experimental Setup

In order to estimate the impact of XML on the coverage of unmodeled defects, we simulated RBFs (see Section III) in the circuits with and without XML. The fault set consisted of 10 000 randomly selected nonfeedback faults (i.e., those that do not introduce asynchronous or combinatorial loops into the circuit), where available. For calculating the RBF coverage FC, we employed the density function ρ derived from one used in [25] (which is based on the data in [13] and assigns lower probability to higher values of bridge resistance).

The RBF model cannot handle unknown values at circuit inputs in a meaningful way because the detection conditions are affected by the input values even if one of the inputs has a controlling value. Hence, we perform a Monte–Carlo simulation of the circuit with and without XML. The X values in the test set IP are set randomly, resulting in a test set IP_1. Resistive bridging fault simulation is performed with test set IP_1 without unknown values. The simulation is repeated 100 times with test sets $\mathrm{IP}_1, \mathrm{IP}_2, \ldots, \mathrm{IP}_{100}$. (All known bits in IP are preserved in every IP_i, and the X values are set randomly.) The average RBF coverage over $\mathrm{IP}_1, \mathrm{IP}_2, \ldots$ is determined then.

Fault detections at some of the output bits should not be accounted for. In absence of an XML, the output bits which are X values do not contribute to detection. We refer to the test setting without an XML as to the *base scenario*, and we denote the output bits with unknown values as $X_{\rm Base}$. If an XML is present,

Procedure Monte_Carlo_Evaluation
Input:
Input Pattern Set IP with X values;
Set X_{Base} of output bits with X values;
For K XMLs, sets $X_1, X_2, \ldots, X_K$ of bits masked out
Output:
Average RBF coverage $RBFC^{\emptyset}_{Base}$ of the base scenario;
For K XMLs, average RBF coverages
$RBFC^{\emptyset}_1, RBFC^{\emptyset}_2, \ldots, RBFC^{\emptyset}_K$
(1) $RBFC^{\emptyset}_{Base} := RBFC^{\emptyset}_1 := \cdots := RBFC^{\emptyset}_K := 0$;
(2) **for** ($i := 1$ **to** 100) **begin**
(3) $IP_i := IP$ with X values randomly assigned to 0s / 1s;
(4) $RBFC^{\emptyset}_{Base} := RBFC^{\emptyset}_{Base} + RBFSim(IP_i, X_{Base})$;
(5) **for** ($j := 1$ **to** K)
(6) $RBFC^{\emptyset}_j := RBFC^{\emptyset}_j + RBFSim(IP_i, X_j)$;
(7) **end for**
(8) **return** $RBFC^{\emptyset}_{Base}, RBFC^{\emptyset}_1, RBFC^{\emptyset}_2, \ldots, RBFC^{\emptyset}_K$
end Monte_Carlo_Evaluation;

Fig. 6. Monte–Carlo estimation of unmodeled defect coverage.

then no detection is possible at the masked bits. Several different architectures of XML are synthesized, using the bit-based and the pattern-based approach and different values of n, and the XML silicon area cost is determined for these architectures. Let the number of these architectures be K, and let X_i be the set of bits masked by the ith XML $1 \leq i \leq j$. (Note that $X_{\rm Base} \subseteq X_i$ always holds).

In order to account for masking, we modified the RBF simulator from [17] such that fault detections by some patterns at some outputs are excluded from consideration. Procedure $\mathrm{RBFSim}(\mathrm{IP}', X')$ simulates the test set IP' (which is not allowed to have X values) not accounting for the detections at the bits specified by X'.

The exact flow of the experiment is shown in Fig. 6. For each of 100 test sets IP_i (which have been obtained from the original test set IP by randomly assigning the X values, Line 3), we perform a total of $K + 1$ simulation runs. The first run (Line 4) determines RBF coverage $\mathrm{RBFC_{Base}}$ for the base scenario (i.e., when the bits with unknown values $X_{\rm Base}$ at the outputs do not contribute to fault detection). The same is repeated for every of the K XML architectures, resulting in RBF coverages $\mathrm{RBFC}_1, RBFC_2, \ldots, RBFC_K$ (Lines 5–7). Note that $\mathrm{RBFC_{Base}}$ is always greater or equal than any RBFC_j. The difference $\mathrm{RBFC_{Base}} - RBFC_j$ is the indicator of the coverage loss for unmodeled defects due to masking out known values by the jth XML. The averaged RBF values (indicated by superscript $\emptyset$) are the output of the experiment (Line 8).

B. Results

Table II summarizes the results for the pattern-based relevant bit selection procedure and and Experiment 1 (X values randomly injected at 1% of the inputs), while Table III contains the results when the bit-based approach has been used. The first three columns give the circuit name, the number "Bits" of bits masked out in the base scenario (which is the number of X values at the output) and "FC," the average global fault coverage FC for the base scenario. The remainder of the table contains the data on XML architecture. For various values of n, the size of synthesized logic in GEs ("LS"), the number of bits masked out ("Bits"), and the average global fault coverage FC ('FC') are reported. For three of the circuits (c3540, c6288 and c7552), *G*-ADI required for calculating FC was not available. For these circuits, *G*-ADI in the denominator is over-approximated by $[0, R_{\max}]$, where $R_{\max}$ is the maximal bridge resistance for which a faulty effect can be produced. Note that by over-approximating the denominator the fault coverage may be below its real value. However, the base scenario and all XML measurements are affected by this to the same extent, so comparing them is still meaningful.

From the table, it can be seen that the logic size does grow with n, however much slower than n. The RBF coverage loss is not dramatic even for $n = 1$, but for $n = 3$ the difference to the base scenario is very small for most circuits. Note that the silicon area cost for $n = 3$ and $n = 1$ is quite similar in most cases.

Results of Experiment 2 with 3% unknown values (only for the bit-based method) are reported in Table IV. The structure of Table IV is identical to Table III. It can be seen that the coverage drop is quite severe for $n = 1$ for some of the circuits. In particular, for c0499 and c1355 the loss is a double-digit number. In such cases, higher values of n are required in order not to lose too much of the unmodeled defect coverage.

Fig. 7 shows the trade-off between the number of masked bits, the logic size and the RBF coverage in graph form for 1% and 3% unknown values.

The results suggest that for low fractions of unknown values the XML synthesis procedure based on stuck-at fault detection is quite effective. Even if no n-detection properties are taken into account ($n = 1$), the RBF coverage loss is small: only for two out of 20 circuits (s5315 and cs38584) in Table III the coverage loss is more than 0.5%. For small $n > 1$, the coverage loss becomes negligible: for $n = 3$, the coverage loss is below 0.2% for all circuits and it is over 0.1% for only three circuits, as opposed to 17 circuits for $n = 1$. But for a higher percentage of X values, preserving n-detection is essential in maintaining the coverage of unmodeled defects.

C. Comparison With Earlier Work

Table V compares our results with those of [6] (industrial circuits not available to us have been used in [7], [8]). We quote the results obtained using the bit-based method for relevant bit selection and $n = 1$, because it corresponds to the goal of [6] (to ensure that every stuck-at fault is detected at least once without considering unmodeled defects or n-detection). Column 2 ("Pat") quotes the number of required test patterns. These patterns are embedded into a sequence of length 10 K. We assume that the other patterns (*irrelevant* in terms of fault detections) from that sequence are masked out completely, as is also done in [6, (section 4.5)]. Column 3 contains the size of XML generated by our approach in GE, *not* including the logic for masking out the irrelevant patterns mentioned above. The synthesis and cost of such logic is highly related to the way deterministic patterns are embedded into the test sets and is beyond the topic of this article. The percentage p of X values among the output bits is shown in the fourth column (it corresponds to p from [6] and is obtained from the data of Table III as (100% · "Base Bits")/("Pat" · "Outs") for the respective circuits).

TABLE II
EXPERIMENTAL RESULTS, PATTERN-BASED RELEVANT BIT SELECTION (1% X VALUES AT THE INPUTS)

Circ	Base		$n=1$			$n=3$			$n=5$			$n=10$		
	Bits	FC	LS	Bits	FC	LS	Bits	FC	LS	Bits	FC	LS	Bits	FC
c0432	28	95.72	23	55	95.69	26	31	95.72	28	30	95.72	30	28	95.72
c0499	63	99.29	37	556	99.29	53	361	99.29	62	310	99.29	75	176	99.29
c0880	20	96.63	25	88	96.37	29	39	96.61	33	30	96.62	35	23	96.63
c1355	128	99.58	62	1277	99.22	86	894	99.39	101	669	99.44	132	521	99.51
c1908	183	99.44	122	881	99.41	168	656	99.42	187	547	99.44	219	394	99.44
c2670	160	97.89	128	2021	97.73	181	1103	97.88	199	746	97.89	243	420	97.89
c3540	205	96.94	157	588	96.91	171	383	96.94	205	322	96.94	212	278	96.94
c5315	406	99.27	228	2657	98.92	365	1485	99.13	428	1107	99.19	494	770	99.27
c6288	143	90.38	51	188	90.25	53	151	90.38	56	143	90.38	56	143	90.38
c7552	602	98.81	358	3561	98.66	468	2056	98.79	528	1633	98.80	582	1100	98.81
cs00298	8	97.48	10	37	97.45	11	18	97.48	11	12	97.48	12	11	97.48
cs00344	11	95.68	13	37	94.60	14	18	95.66	14	14	95.66	15	11	95.68
cs00400	17	98.28	26	75	98.19	27	43	98.27	28	37	98.24	30	29	98.28
cs00444	9	97.82	13	66	97.76	16	35	97.82	18	34	97.82	20	17	97.82
cs00526	23	98.35	35	180	98.29	45	96	98.34	48	77	98.35	49	54	98.35
cs00713	15	98.68	20	135	98.57	27	69	98.67	29	48	98.67	31	25	98.68
cs05378	2024	98.97	455	8292	98.84	608	5262	98.94	716	4410	98.95	863	3320	98.96
cs13207	2808	99.10	1648	102315	99.04	2364	61531	99.08	2796	46632	99.09	3461	26858	99.10
cs15850	2115	98.74	1540	32681	98.58	2059	18422	98.69	2328	13832	98.72	2895	8363	98.73
cs38584	5626	96.47	3746	84430	96.14	5535	46974	96.38	6524	33201	96.43	7917	20763	96.45

TABLE III
EXPERIMENTAL RESULTS, BIT-BASED RELEVANT BIT SELECTION (1% X VALUES AT THE INPUTS)

Circ	Base		$n=1$			$n=3$			$n=5$			$n=10$			$n=15$			$n=20$		
	Bits	FC	LS	Bits	FC	LS	Bits	FC	LS	Bits	FC	LS	Bits	FC	LS	Bits	FC	LS	Bits	FC
c0432	28	95.72	21	66	95.58	24	50	95.68	25	41	95.71	26	31	95.72	28	29	95.72	28	29	95.72
c0499	63	99.29	35	586	99.29	51	347	99.29	58	293	99.29	73	165	99.29	85	117	99.29	99	84	99.29
c0880	20	96.63	23	110	96.35	30	36	96.62	32	30	96.62	34	23	96.63	35	23	96.63	36	21	96.63
c1355	128	99.58	60	1371	99.26	83	979	99.40	97	832	99.40	129	535	99.47	142	431	99.52	166	341	99.53
c1908	183	99.44	110	1215	99.26	139	920	99.40	157	646	99.43	197	531	99.44	225	440	99.44	232	344	99.44
c2670	160	97.89	112	2690	97.67	156	1360	97.85	185	1021	97.88	226	682	97.89	247	490	97.89	262	344	97.89
c3540	205	96.94	118	1080	96.64	149	580	96.92	169	487	96.92	191	403	96.93	214	287	96.94	225	269	96.94
c5315	406	99.27	206	3430	98.69	330	2021	99.15	369	1502	99.10	451	1103	99.24	467	1096	99.23	496	979	99.20
c6288	143	90.38	49	211	90.23	52	159	90.36	54	144	90.38	56	143	90.38	56	143	90.38	56	143	90.38
c7552	602	98.81	336	4359	98.52	437	3044	98.77	493	2225	98.80	550	1713	98.80	595	1283	98.80	610	1040	98.80
cs00298	8	97.48	9	36	97.45	11	13	97.48	11	15	97.48	12	10	97.48	13	8	97.48	13	8	97.48
cs00344	11	95.68	12	65	95.34	13	22	95.64	14	20	95.68	15	11	95.68	15	11	95.68	15	11	95.68
cs00400	17	98.28	25	117	97.91	27	52	98.27	27	45	98.27	28	29	98.28	32	22	98.28	33	17	98.28
cs00444	9	97.82	13	97	97.71	15	50	97.82	16	40	97.82	20	16	97.82	21	14	97.82	22	10	97.82
cs00526	23	98.35	33	281	98.09	41	166	98.32	44	97	98.34	48	58	98.35	50	43	98.35	51	37	98.35
cs00713	15	98.68	18	154	98.58	24	84	98.66	27	60	98.68	31	24	98.68	31	23	98.68	32	16	98.68
cs05378	2024	98.97	338	10383	98.63	470	7810	98.87	555	6556	98.91	703	5170	98.96	801	4448	98.97	883	4032	98.97
cs13207	2808	99.10	1351	113718	98.96	2082	70397	99.07	2604	56385	99.09	3333	35440	99.10	3911	23522	99.10	4291	17866	99.09
cs15850	2115	98.74	1164	40005	98.44	1807	22553	98.67	2161	15609	98.68	2775	9264	98.71	3120	7181	98.73	3275	6014	98.73
cs38584	5626	96.47	3286	97955	95.42	5145	53161	96.33	6177	37775	96.41	7724	22766	96.43	8490	16409	96.45	9050	13590	96.45

Column 5 ("SeqL") of Table V contains the length of the sequence used in [6], including those test patterns that are masked out completely using the technique from Section 4.5 in that paper; hence, the comparability with column "Pat" is limited. In column 6, stuck-at fault coverage achieved by the patterns used in [6] is quoted (in contrast to these numbers, the fault efficiency of the patterns employed in this work is always 100%).

The remainder of Table V summarizes the silicon area cost of the architecture from [6] that should be compared with the numbers in the third column. [6] reports results for $p = 0.05\%$, 0.1% and $p = 0.2\%$, where p is the percentage of the output values set to X randomly. Note that we set the *input values* to X with a probability larger than 0.2% and thus end up with more X values at the outputs, which are also correlated in a realistic way (their percentage is quoted in column 4). For each p, the number S of seeds and the number P of stages in the LFSR is quoted in [6]. We assume that the logic size of the overall architecture from [6] in gate equivalent is calculated according to the formula

$$\mathrm{GE} = 6 \cdot P + 2 + S \cdot \frac{P}{4}. \tag{3}$$

We count a flip-flop as six GE: two gates for the RS circuit, three gates for the multiplexer, and one gate for edge handling. We assume that there are two XOR gates to implement feedback, and we count an XOR gate as one GE, which is an under-approximation. Hence, the LFSR totals $6 \cdot P + 2$ GE. We use the number P and not the higher number F in calculation in order to reflect the use of the technique from [6, Section 4.5]. Note that the LFSR is *not* used for random pattern generation; it is a resource present exclusively for the purpose of masking X values. $S \cdot P$ bits have to be stored on-chip (reseeding information); we assume a PLA implementation and count one bit as 1/4 GE. We neglect the control logic for loading seeds from the PLA into

TABLE IV
EXPERIMENTAL RESULTS, BIT-BASED RELEVANT BIT SELECTION (3% X VALUES AT THE INPUTS)

Circ	Base		$n=1$			$n=3$			$n=5$			$n=10$			$n=15$			$n=20$		
	Bits	FC	LS	Bits	FC	LS	Bits	FC	LS	Bits	FC	LS	Bits	FC	LS	Bits	FC	LS	Bits	FC
c0432	61	94.01	33	122	93.20	41	88	93.86	43	78	94.00	45	70	94.01	46	66	94.01	47	63	94.01
c0499	314	94.53	75	1222	43.27	111	1023	85.57	123	912	88.20	183	703	88.59	231	525	94.10	275	405	94.52
c0880	63	96.34	51	240	95.63	64	110	96.25	67	117	96.28	81	81	96.34	83	71	96.34	83	66	96.34
c1355	542	95.10	98	2245	70.56	156	2110	73.89	203	1859	76.22	258	1581	81.80	304	1287	88.06	411	1047	89.24
c1908	360	99.10	136	1757	98.61	208	1261	99.01	249	1132	99.08	300	888	99.10	333	776	99.10	356	674	99.10
c2670	456	93.98	207	4477	92.24	286	3084	93.80	358	2176	93.92	440	1453	93.97	507	1205	93.98	541	774	93.98
c3540	433	96.18	219	1503	95.06	273	1120	95.80	314	930	96.08	381	735	96.15	414	640	96.17	419	564	96.17
c5315	925	98.81	305	5090	96.74	502	3820	98.14	623	3153	98.22	762	2238	98.51	846	1976	98.67	875	1678	98.73
c6288	326	88.25	72	427	85.74	77	344	88.12	82	333	88.25	89	327	88.25	89	326	88.25	89	326	88.25
c7552	1574	97.57	532	6801	95.14	792	5125	97.28	880	4866	97.45	1004	3984	97.50	1071	3325	97.53	1152	2409	97.53
cs00298	29	96.94	29	133	96.35	33	76	96.90	38	45	96.93	43	39	96.94	44	33	96.94	46	30	96.94
cs00344	20	95.69	21	71	94.75	26	33	95.67	29	27	95.69	30	20	95.69	30	20	95.69	30	20	95.69
cs00400	49	97.58	38	253	95.97	53	155	97.36	57	123	97.46	65	60	97.58	70	54	97.58	74	50	97.58
cs00444	41	97.01	36	191	95.60	43	89	96.88	48	86	96.87	53	49	97.00	54	45	97.01	54	42	97.01
cs00526	94	98.19	77	493	97.72	99	300	98.08	104	235	98.14	109	218	98.17	132	146	98.19	134	129	98.19
cs00713	87	98.31	57	447	97.74	84	273	98.19	99	174	98.30	105	126	98.31	107	123	98.31	111	99	98.31
cs05378	2926	98.54	518	16112	97.64	792	13659	98.35	997	11497	98.38	1299	8700	98.50	1484	7034	98.53	1658	6331	98.53
cs13207	8536	98.93	2041	166074	97.76	3751	131954	98.20	4874	113199	98.25	6988	81267	98.29	8397	55940	98.30	9531	43721	98.30
cs15850	5872	98.38	1786	64684	96.22	3135	46191	97.34	4056	36472	97.57	5539	23249	97.74	6527	17229	97.74	7113	14826	97.75
cs38584	12525	95.95	5053	149702	94.14	8766	96010	95.69	11078	74158	95.82	14586	45301	95.92	16423	32520	95.92	17464	27853	95.90

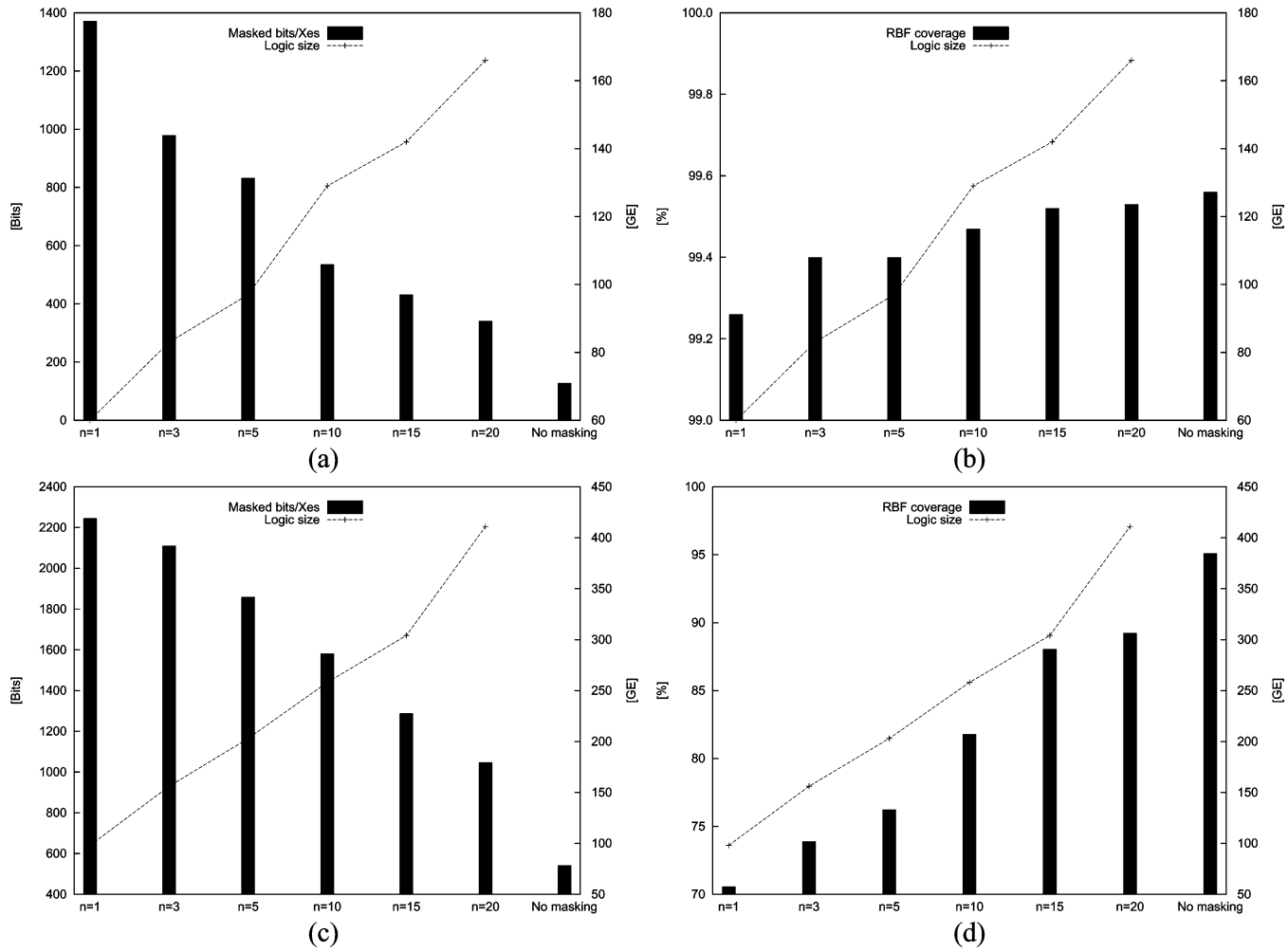

Fig. 7. Results for c1355. (a) Number of masked bits and logic size as function of n (1% Xs). (b) RBF coverage and logic size as function of n (1% Xs). (c) Number of masked bits and logic size as function of n (3% Xs). (d) RBF coverage and logic size as function of n (3% Xs).

TABLE V
RESULT COMPARISON TO [6]

Circ	Proposed			Naruse et al., ITC 2003 [6]										
	Pat	Si area cost, GE	p	SeqL	FC	Silicon area cost								
						$p = 0.05\%$			$p = 0.1\%$			$p = 0.2\%$		
						S	P	GE	S	P	GE	S	P	GE
cs298	25	9	1.6%	600	100	1	6	39.5	2	6	41	2	14	93
cs344	16	12	2.6%	300	100	1	6	39.5	1	8	52	1	20	127
cs400	28	25	2.2%	500	98.7	1	9	58.25	1	19	120.75	1	19	120.75
cs444	28	13	1.2%	600	97.3	1	10	64.5	1	8	52	4	10	72
cs526	55	33	1.5%	2600	97.7	1	20	127	8	11	90	4	30	212
cs713	31	18	1.2%	2500	91.6	1	20	127	4	30	212	9	26	216.5
cs5378	121	338	7.3%	10000	98.0	55	19	377.25	59	27	562.25	103	30	954.5
cs13207	276	1351	1.3%	42000	97.6	278	30	2267	378	28	2816	462	31	3768.5
cs15850	139	1164	2.2%	1200	97.0	40	30	482	60	28	590	100	28	870
cs38584	147	3286	2.2%	30000	95.3	571	20	2977	692	28	5014	1002	21	5388.5

the LFSR and also the logic needed to implement the technique from [6, Section 4.5].

The remaining columns of Table V contain the values of S and P from [6] and the size of the logic in GE estimated using (3). It can be seen that our solution most often requires less silicon area cost despite a higher value of p. Note that the results obtained by the method from [6] might be improved by considering X values correlated in a realistic way (which is done in this work by injecting unknown values at the inputs). However, such results are not available for that method.

V. CONCLUSION

Logic blocks that produce unknown values at their outputs are hard to deal with in a BIST environment, as the signature may be corrupted by the unknown values. Masking the X values at the outputs of such modules allows the use of arbitrary TREs, including those vulnerable to X values. Since most faults are detected by many patterns, some known bits can also be masked without loss of stuck-at fault coverage.

We proposed a method to synthesize XML that works for combinational, sequential, scan and partial scan circuits. It can be integrated into any BIST architecture. While previous works concentrated on sustaining the stuck-at coverage after masking, we are using more conservative metrics based on n-detection, in order to preserve the coverage of unmodeled defects. To the best of our knowledge, this is the first study that considers the effects of X-masking on unmodeled defects. We estimated the coverage of unmodeled defects using a sophisticated RBF model, which accounts for pattern dependency. By varying n, there is a trade-off between the size of the synthesized XML and the coverage of unmodeled defects. Relatively small values of n were sufficient to achieve practically the same coverage as with no masking logic, as long as the fraction of X values to be masked was relatively low. For a higher percentage of X values, sacrificing the n-detection properties of the test set for the sake of minimizing XML results in a significant drop in coverage of unmodeled defects. In such cases, XML architectures synthesized using a high value of n should be used.

ACKNOWLEDGMENT

The authors are grateful to V. Gherman who provided additional data for this paper.

REFERENCES

[1] H.-J. Wunderlich, "BIST for systems-on-a-chip," *INTEGRATION, the VLSI J.*, vol. 26, no. 12, pp. 55–78, Dec. 1998.
[2] A. J. van der Goor, *Testing Semiconductors Memories Theory and Practice*. New York: Wiley, 1998.
[3] S. Mitra and K. S. Kim, "X-Compact: An efficient response compaction technique for test cost reduction," in *Proc. Int. Test Conf.*, 2002, pp. 311–320.
[4] J. Rajski, C. Wang, J. Tyszer, and S. M. Reddy, "Convolutional compaction of test responses," in *Proc. Int. Test Conf.*, 2003, pp. 745–754.
[5] I. Pomeranz, S. Kundu, and S. M. Reddy, "On output response compression in the presence compression in the response of unknown output values," in *Proc. Des. Autom. Conf.*, 2002, pp. 255–258.
[6] M. Naruse, I. Pomeranz, S. M. Reddy, and S. Kundu, "On-chip compression of output responses with unknown values using LFSR reseeding," in *Proc. Int. Test Conf.*, 2003, pp. 1060–1068.
[7] E. H. Volkerink and S. Mitra, "Response compaction with any number of unknowns using a new LFSR architecture," in *Proc. Des. Autom. Conf.*, 2005, pp. 117–122.
[8] T. Clouqueur, K. Zarrineh, K. K. Saluja, and H. Fujiwara, "Design and analysis of multiple weight linear compactors of responses containing unknown values," presented at the Int. Test Conf., Austin, TX, 2005.
[9] M. Arai, S. Fukumoto, and K. Iwasaki, "Analysis of error-masking and X-masking probabilities for convolutional compactors," presented at the Int. Test Conf., Austin, TX, 2005.
[10] S. C. Ma, P. Franco, and E. J. McCluskey, "An experimental chip to evaluate test techniques experimental results," in *Proc. Int. Test Conf.*, 1995, pp. 663–672.
[11] S. M. Reddy, I. Pomeranz, and S. Kajihara, "Compact test sets for high defect coverage," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 16, no. 8, pp. 923–930, Aug. 1997.
[12] B. Benware, C. Schuermyer, S. Ranganathan, R. Madge, P. Krishnamurty, N. Tamarapalli, K. H. Tsai, and J. Rajski, "Impact of multiple-detect test patterns on product quality," in *Proc. Int. Test Conf.*, 2003, pp. 1031–1040.
[13] R. Rodríguez-Montañés, E. M. J. G. Bruls, and J. Figueras, "Bridging defects resistance measurements in a CMOS process," in *Proc. Int. Test Conf.*, 1992, pp. 892–899.
[14] M. Renovell, P. Huc, and Y. Bertrand, "CMOS bridge fault modeling," in *Proc. VLSI Test Symp.*, 1994, pp. 392–397.
[15] ——, "The concept of resistance interval: A new parametric model for resistive bridging fault," in *Proc. VLSI Test Symp.*, 1995, pp. 184–189.
[16] **<PLEASE PROVIDE JOURNAL MONTH AND ISSUE NUMBER.>** M. Renovell, F. Azaïs, and Y. Bertrand, "Detection of defects using fault model oriented test sequences," *J. Electron. Testing: Theory Appl.*, vol. 14, pp. 13–22, 1999.
[17] P. Engelke, I. Polian, M. Renovell, and B. Becker, "Simulating resistive bridging and stuck-at faults," in *Proc. Int. Test Conf.*, 2003, pp. 1051–1059.
[18] H.-J. Wunderlich and G. Kiefer, "Bit-flipping BIST," in *Proc. Int. Conf. Comput.-Aided Des.*, 1996, pp. 337–343.
[19] G. Kiefer, H. Vranken, E. J. Marinissen, and H.-J. Wunderlich, "Application of deterministic logic BIST on industrial circuits," in *Proc. Int. Test Conf.*, 2000, pp. 105–114.
[20] N. A. Touba and E. J. McCluskey, "Altering a pseudo-random bit sequence for scan based BIST," in *Proc. Int. Test Conf.*, 1996, pp. 649–658.

[21] R. K. Brayton, R. Rudell, A. L. Sangiovanni-Vincentelli, and A. R. Wang, "MIS: A multiple-level logic optimization system," *IEEE Trans. Comput.*, vol. C-6, no. 6, pp. 1062–1081, Jun. 1987.
[22] F. Brglez and H. Fujiwara, "A neutral netlist of 10 combinational circuits and a target translator in fortran," in *Proc. Int. Symp. Circuits Syst., Special Sess. ATPG Fault Simulation*, 1985, pp. 663–698.
[23] F. Brglez, D. Bryan, and K. Kozminski, "Combinational profiles of sequential benchmark circuits," in *Proc. Int. Symp. Circuits Syst.*, 1989, pp. 1929–1934.
[24] V. Gherman, H. J. Wunderlich, H. Vranken, F. Hapke, M. Wittke, and M. Garbers, "Efficient pattern mapping for deterministic logic BIST," in *Proc. Int. Test Conf.*, 2004, pp. 48–56.
[25] C. Lee and D. M. H. Walker, "PROBE: A PPSFP simulator for resistive bridging faults," in *Proc. VLSI Test Symp.*, 2000, pp. 105–110.

Yuyi Tang received the B.S. degree from Shanghai Jiaotong University, Shanghai, China, in 1999, and the M.S. degree in electrical engineering from Dresden University of Technology, Dresden, Germany, in 2002.

He continued his research work on VLSI design and design for test at the University of Stuttgart, Germany, where he engaged in the project AZTEKE, supported by German Ministry of Research and Technology (BMBF). His research interests include advanced ASIC design, verification, and test methods.

Hans-Joachim Wunderlich (A'86) received the Dipl.-Math. degree in mathematics from the University of Freiburg, Freiburg, Germany, in 1981, and the Dr.Rer.Nat. (Ph. D.) degree in computer science from the University of Karlsruhe, Karlsruhe, Germany, in 1986, where he became an Assistant Professor in 1990.

From 1991 to 1996, he was a Full Professor for computer science at the University of Siegen. He has been with the University of Stuttgart, Stuttgart, Germany, since 1996, where he became the Director of the Institut für Technische Informatik (Institute of Computer Engineering and Computer Architecture) in 1999, and is currently a Full Professor. He has been a lecturer of tutorials for Europractice and the IEEE Test Technology Educational Program (TTEP) at major conferences including ITC, VTS, and ETW. He is an author or co-author of 4 books and more than 100 papers in the field of test, built-in self test (BIST), and fault tolerance. He and his group takes part in various national and international research projects funded by the German Ministry of Research and Technology (BMBF), by the German Research Association (DFG), by the European Union, and Industrial Partners (ATMEL, IBM, Infineon, and Philips).

Piet Engelke (S'03) received the Diploma degree in computer science from the University of Freiburg, Germany, in 2002. He is currently pursuing the Ph.D. degree and continuing his research on resistive bridging faults.

He spent six months in the Design For Test Group of the Mentor Graphics Corporation, Wilsonville, OR. His research interests include defect based-testing and fault simulation.

Ilia Polian (M'04) received the M.S. degree and the Ph.D. degree (with distinction) in computer science from the Albert-Ludwigs-University, Freiburg, Germany, in 1999 and 2003, respectively.

He is currently a senior member of the scientific staff at the Chair of Computer Architecture, Albert-Ludwigs-University, Germany. His previous affiliations were with Micronas, Freiburg, Germany, IBM Germany R&D, Böblingen, and NAIST, Nara, Japan. His research interests include defect modeling, design for testability, and formal verification of hybrid and real-time systems.

Dr. Polian was European Champion and Vice World Champion at the 1999 ACM International Collegiate Programming Contest, VDE Award Laureate, 1999, and Wolfgang-Gentner Award Laureate, 2005.

Bernd Becker (SM'02) studied mathematics and computer science at the University of Saarland, Saarbrücken, Germany, from 1973 to 1982.

Between 1979 and 1988, he was with the Sonderforschungsbereich Electronic Speech Recognition (1979–1981), Institute for Computer Science and Applied Mathematics (1981–1983), and the Sonderforschungsbereich VLSI Design Methods and Parallelism (1984–1988), all at the University of Saarland. From 1989 to 1995, he was an Associate Professor for Complexity Theory and Efficient Algorithms at the J. W. Goethe-University Frankfurt. In 1995, he joined the Faculty of Applied Sciences at the Albert-Ludwigs-University Freiburg as a Full Professor (Chair of Computer Architecture). His research interests include data structures and efficient algorithms (for circuit design), design, test, and verification of circuits and systems, multimedia in research, and teaching.

Jürgen Schlöffel <PLEASE PROVIDE BETTER QUALITY AUTHOR PHOTO.> (M'04) received the Diploma degree in physics from Georg-August-University, Goettingen, Germany, in 1986.

He is Project Manager and Senior Engineer at the Design Technology Center, Philips Semiconductors, Hamburg, Germany. His interests include advanced testing techniques, failure diagnosis, and design automation for DSM technologies. He is a member of VDE, board member of the ITG working group for test and reliability, and steering group member of the German Edacentrum.

Friedrich Hapke <PLEASE PROVIDE BETTER QUALITY AUTHOR PHOTO.> received the Diploma degree in electronics from the Fachhochschule, Hamburg, Germany.

He is the Design For Test Manager of the Chief Technology Office, Philips Semiconductors, Hamburg, Germany. His interests include advanced testing techniques, automatic test pattern generation, failure diagnosis, and design automation in general for DSM technologies. He holds over ten patents in the area of Design For Test inventions. He is a member of the Supervisory Board of the German Edacentrum.

Michael Wittke <PLEASE PROVIDE BETTER QUALITY AUTHOR PHOTO.> received the Diploma degree in computer science from the University of Karlsruhe (TH), Karlsruhe, Germany, in 1987.

He is Senior Engineer at the Design Technology Center, Philips Semiconductors, Hamburg, Germany. His interests are ATPG, test compression, and logic BIST.