Scalable Selector Architecture for X-Tolerant Deterministic BIST

Peter Wohl
Synopsys Inc.
Williston VT 05495
wohl@synopsys.com

John A. Waicukauski
Synopsys Inc.
Tualatin OR 97062
johnwaic@synopsys.com

Sanjay Patel
Synopsys Inc.
Beaverton OR 97006
sanjayp@synopsys.com

Abstract

X-tolerant deterministic BIST (XDBIST) was recently presented as a method to efficiently compress and apply scan patterns generated by automatic test pattern generation (ATPG) in a logic built-in self-test architecture. In this paper we introduce a novel selector architecture that allows arbitrary compression ratios, scales to any number of scan chains and minimizes area overhead. XDBIST test-coverage, full X-tolerance and scan-based diagnosis ability are preserved and are the same as deterministic scan-ATPG.

Categories and Subject Descriptors: B.8.1 **[Performance and Reliability]:** Reliability, Testing and Fault-Tolerance.

General Terms: Algorithms, Design.

Keywords: test-data compression, test-generation (ATPG).

1. Introduction

Increases in size, complexity and operating speed of digital designs contribute to even faster increases in test costs because test data volume, test application time and test-equipment pin counts increase [1]. Scan [2], [3] and logic BIST [4] have been the fundamental design-for-test (DFT) methods used to control test cost: scan enables high test quality through high-coverage structured test, and BIST reduces test equipment cost by reducing test data volume. Most data and test time in scan-based test are in scan-chain load/unloads. Load data can be reduced by orders of magnitude by loading the scan chains of the design under test (DUT) from an on-chip pseudo-random patterns generator (PRPG) and re-seeding the PRPG to obtain the desired "care bits" (scan cells that must be set to a certain value) for a deterministic test [5], [6], [7]. Unload data can also be reduced by orders of magnitude by compressing unloaded values into a signature analyzer, often implemented as a multiple-input signature register (MISR) [8], [9]. Once test data is greatly reduced, test application time can be shortened by organizing DUT scan cells into a large number of short chains that can be loaded and unloaded in a small number of cycles, and by adding on-chip compression/decompression circuitry to move data to/from the tester.

However, one of the main obstacles to widespread adoption of logic BIST is that the DUT must be modified to prevent unknown (X) values from propagating to the MISR. A single X would compromise the self-test by making the fault-free MISR signature unpredictable [10]. The design under test can contain X-sources such as buses, non-scan state elements, embedded memories, timing-sensitive paths, etc. If all X-sources can be detected through pre-silicon analysis and the design can be modified to avoid X's from propagating to the scan chains, then a MISR-based signature analyzer can be successfully used [6]. However, in some cases, such as race conditions, X sources may not be known before silicon. In other cases, DUT modifications to avoid X propagation are undesired for timing reasons. Therefore, recent efforts have focused on devising schemes that combine high-coverage scan-based deterministic tests with BIST-based test data and time reduction, while allowing for X values in the output response of the DUT. Section 2 overviews output compression techniques, section 3 introduces a novel scanout compression architecture, section 4 presents results, and section 5 concludes this paper.

DAC 2004, June 7-11, 2004, San Diego, California, USA

2. X-tolerant Compression of Unload Data

Several methods have been devised to reduce test data volume, maintain high test coverage and, at the same time, tolerate X-values: masking, compaction and selective observation of unload values.

In the first category, circuitry is added to mask selected unload values so that X's do not reach the MISR [7], [11], [12], [13]. However, if the number of X's is too large or the distribution is unfavorable, masking may have a high area overhead or it may not be selective enough; therefore, non-X values would also be masked out, resulting in a potential loss of coverage or increased number of patterns. Also, if unexpected X's are detected, the test patterns and the masking hardware may need to be regenerated to provide new X-masks.

A second category of methods achieves compression by compacting unload data and allowing the tester to selectively ignore measures [14], [15]. However, design and fault-independent test-response compaction necessarily results in some loss of data; therefore, aliasing and X-masking can occur. Aliasing results when multiple error values mask each other; X-masking results when unknown values mask error values from observation at a compactor output. For efficiency, fault simulators do not simulate the compactor; instead, when a fault effect propagates to a scan cell the fault is considered detected. Thus, aliasing and X-masking may reduce effective test quality because faults may be undetected even though credit for detection has been taken. Compaction may also reduce fault-diagnosis resolution because the error inputs cannot always be uniquely identified. Scan-chain failures or failures on global nets such as clocks or scan-enable result in a large number of fails and possibly X-values so that accurate diagnosis is impossible unless additional test applications unload un-compacted data [16].

Finally, XDBIST relies on selective observation of unload values to achieve significant reduction in both test data and tester cycles [17]. XDBIST has full X-tolerance and ensures the same high test coverage and diagnosability as deterministic scan-ATPG (where all scan cells can be controlled and observed). The DUT scan chains are reconfigured into a large number of short chains and connected to an on-chip

decompressor — the observe selector (Figure 1). Test patterns are generated by a modified deterministic ATPG. Load values are compressed into PRPG seeds as part of test generation so that all care bits are set to the desired values [6], [17]. A small percentage of scan chains are selected to be unloaded during each pattern and the tester directly compares unloaded values. The process of selecting the scan chains to observe each pattern and compressing selection control data is integrated with test generation. XDBIST can tolerate any number of X's propagating to the scan chains, with no degradation in compression or application efficiency. As with traditional scan-ATPG, unexpected X's can be masked off at the tester on a per-cell basis using the same pattern set. Masking X measures never results in unwanted masking of non-X values. To avoid adding cycles for initializing the PRPG, the PRPG-LFSR is coupled with a PRPG shadow that allows re-seeding with 0-cycle overhead [6]. A similar shadow is used to load the selector control bits [17], so no extra cycles are added. The tester views the DUT as a traditional scan-based design. However, when compared to scan-ATPG, the design with XDBIST requires fewer shift cycles per load and can have fewer scan-in and scan-out pins (Figure 1), which significantly reduces test data and tester cycles. XDBIST patterns fit directly into a scan-test flow for failure diagnosis [17].

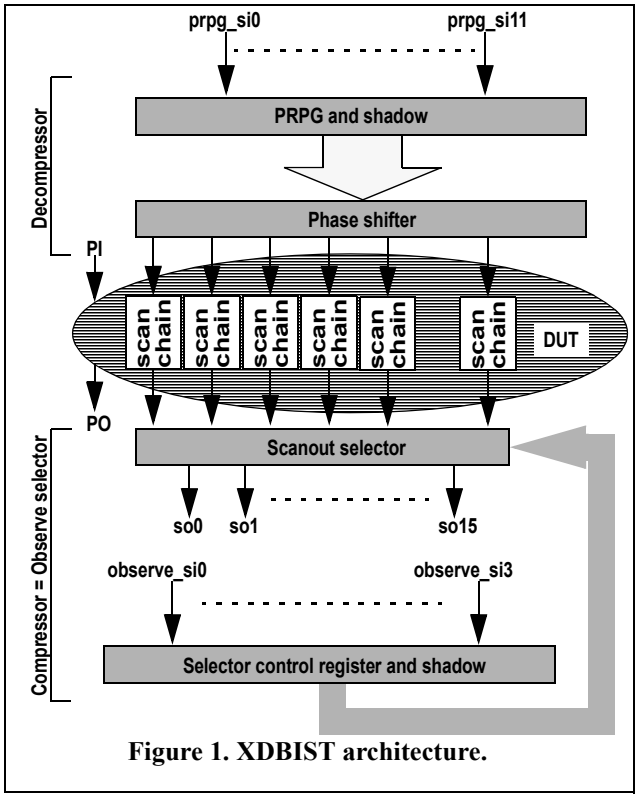

Figure 1. XDBIST architecture.

The scanout selector architecture is key to the effectiveness of XDBIST test generation, compression and area overhead. This paper introduces a new selector architecture that, when compared to the selector of [17], allows arbitrary compression ratios, scales to any number of scan chains, minimizes area overhead and reduces selector control data.

3. Scanout Selector Design

To minimize XDBIST data and tester cycles, the number of patterns must be minimal. The ability to unload any desired set of chains in every pattern affects the pattern count. Therefore, the scanout selector (Figure 1) must ensure a high probability of being able to observe any desired subset of scan chains. Unfortunately, a fully connected scanout selector (every scan chain connects to every scanout pin so that any subset of chains can be observed) is impractical because of its high overhead. An optimal scanout selector needs to offer a reasonable trade-off between multiple requirements such as the ability to observe desired chains, area, delay, wiring, control data size, scalability, etc. A previous approach to designing a selector that meets most of these criteria was presented in [17]. The design goals for the scanout selector presented in this paper include significantly improved characteristics:

(1) Any number of scan chains can be supported (vs. up to 512 [17]).
(2) The compression ratio is selectable (vs. fixed 32:1 [17]).
(3) Area overhead and wiring are reduced, most significantly for small designs.
(4) Maximal delay selection from a chain output to a chip pad is possible (vs. fixed [17]).
(5) Control data size is reduced.
(6) High observation probability of multiple chains is ensured so that the pattern count is not inflated.
(7) Full X-tolerance and full scan-based diagnosability is maintained.

3.1. Simple Selector

The first design goal, scalability to any number of scan chains, is most easily achieved by the smallest and simplest scanout selector: for a C:1 compression ratio, groups of C scan chains are connected through a multiplexor to a scanout pin (C=32 in Figure 2).

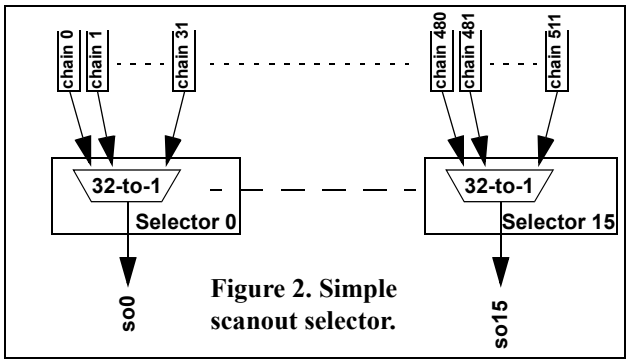

Figure 2. Simple scanout selector.

The area and timing impact of this selector are minimal, as shown in the "Simple selector" column, Table 1, for two cases: 32:1 and 64:1 compressions. The delay is computed as the number of 2:1 MUXes on the path from a scan chain to a scanout pin. The wiring overhead is estimated as the *maximum fanin* of any module (i.e., a 32:1 MUX has fanin 37 = 32 data inputs + 5 select inputs) and the *average fanout* of scan chains. The next row in Table 1 shows the number of control bits needed for each output: five for a simple 32:1 selector and six for a simple 64:1 selector. Total area is shown next for three sample designs with 512, 128 and, respectively, 64 scan chains; area numbers are based on normalized equivalent 2-input gates: each AND, NAND, OR, NOR and TSD is equivalent to one gate, each XOR, XNOR and 2:1 MUX is equivalent to three gates, and a DFF is equivalent to six gates.

The simple selector scales to any number of scan chains and can support any compression ratio, but restricts the subsets of scan chains that can be simultaneously observed: for example (Figure 2), selecting chain 0 implies that chains 1, 2,.., 31 cannot be observed in the same pattern. To evaluate each selector's performance (and thus the efficiency of XDBIST patterns), we determined the probability of suc-

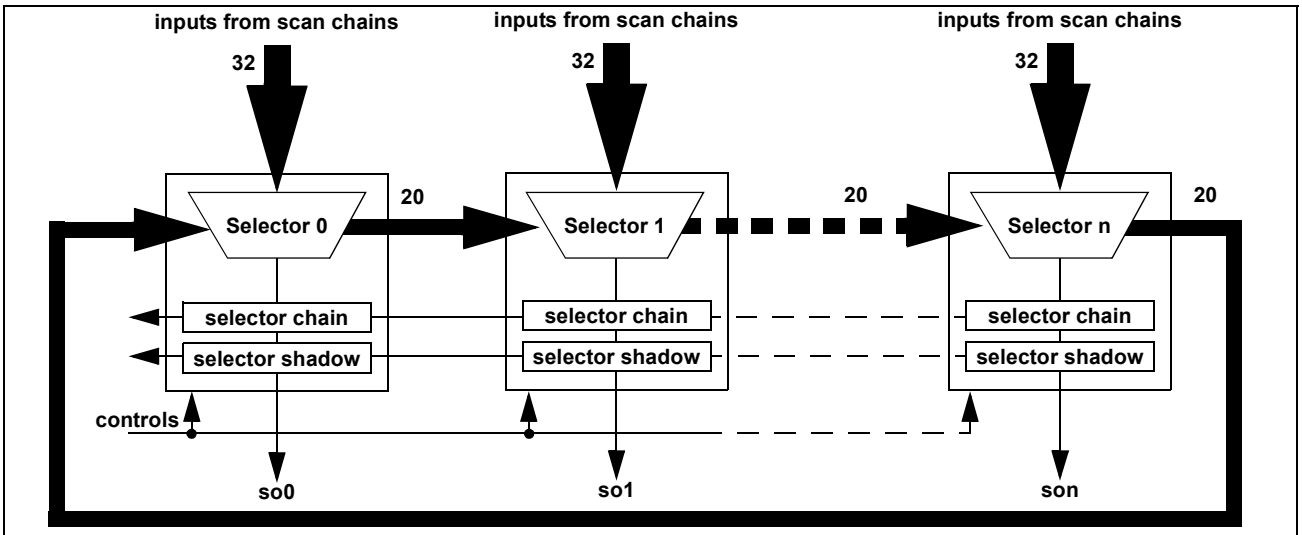

Figure 3. Scalable selector architecture.

cessfully observing all chains in a randomly selected subset of chains. This probability can be derived analytically or experimentally. For the latter, we generated 1000 random selections and determined whether the selected chains could be routed to scanout pins. The "Simple selector" column in Table 1 shows a few data points: one chain can always be observed, but the probability to observe two chains with two outputs is only 48%, eight chains with 10 outputs is only 2% and eight chains with 16 outputs is less than 25%. To reduce pattern count and CPU time, ATPG must efficiently generate patterns that maximize fault detection. At the beginning of test generation, every pattern can detect a very large number of faults by observing almost any full set of scan chains. The latter part of test generation focuses on hard-to-detect faults, which account for most patterns. During this stage, merging as many faults as possible into few patterns requires that a given, small set of chains be observed; not being able to route the selected chains to outputs would increase pattern count because hard-to-detect faults typically offer few observation opportunities. Unfortunately, the low observation probabilities of the simple selector renders it unsuitable for a compact pattern set.

3.2. Modular Selector

The selector presented in [17] meets and even exceeds observation probabilities, and has low delay and wiring overhead ("Modular selector" column in Table 1). However, it can only support a maximum of 512 chains and a fixed 32:1 compression ratio. It also requires twice as many control bits per output than the simple selector, which results in increased area and input data volume. Most importantly, its area overhead is large, especially for small designs that populate only 128 or even 64 out of the 512 supported scan chains.

3.3. The Proposed Scalable Selector

To meet the design goals outlined at the beginning of section 3, we started with the simple scanout selector architecture (Figure 2) and added side connections to allow each scan-chain to be observed at more than one output (Figure 3). Each output (so0, so1,..., son) is connected to an identical selector. Each selector has 32 inputs from scan chains, 20 side inputs from its neighboring selector to the left, and 20 side outputs to its neighboring selector to the right, in a circular configuration. The selector chains and the selector shadow chains are linked together. As with the simple selector, chains 0...31 are connected to selector 0, the next 32 chains to selector 1, etc. To maximize the probability that the scan chains chosen during ATPG can be routed to outputs, scan chains having the same capture clock are connected to different selectors.

Unlike the simple selector, selecting chain 0 for output so0 does not block chain 1 to be simultaneously observed; indeed, chain 1 can be routed, through the side connections to selector 1 and observed at output so1. At the same time, chain 2 could be routed through selector 1, into selector 2 and observed at so2, etc. The side connections form topological loops so that any set of chains has the same observation opportunity, irrespective of the selector to which it is connected. However, the controls to the selectors, computed as part of the test generation process, ensure that no loop is ever sensitized; i.e., no signal can ever propagate around all selectors. Moreover, a user-controllable parameter, "max_side_routing", determines how many selectors a scan chain can be routed through before reaching an output pin. For instance, if max_side_routing = 5 then up to five chains connected to the same selector can be simultaneously observed. If the value of max_side_routing is set too high, some chains may encounter too long a delay and the unload data cannot be measured at the selected output pin in the desired tester cycle time. If the value of max_side_routing is too low, the probability of observing a desired subset of chains becomes too low. Setting max_side_routing to one reduces the function of the scalable selector to that of a simple selector. The probabilities shown in the "Scalable selector" column, Table 1, were obtained with max_side_routing = 5.

The scalable selector architecture (Figure 3) is conceptually very versatile and meets most design goals outlined at the beginning of section 3, but additional implementation trade-offs must be made to reduce area and control data size. Figure 4 shows a relatively straightforward implementation of a scalable selector that allows up to four chains to be routed to the next selector for observation, which is equivalent to max_side_routing = 5. The main 36-to-1 multiplexor can select any of the 32 scan chains connected to the selector or any of the four side inputs from the previous selector. The four 32-to-1 selectors can select up to four values to be routed to the next selector. To keep the four MUXes to only 32 inputs (five control bits), it is sufficient to connect each MUX to a single input from the previous selector and a

unique subset of 31 out of the 32 scan chains. Unfortunately, even with these savings, an implementation of the selector in Figure 4 would be five times larger than the simple selector (Figure 2) and would require five times as many control bits!

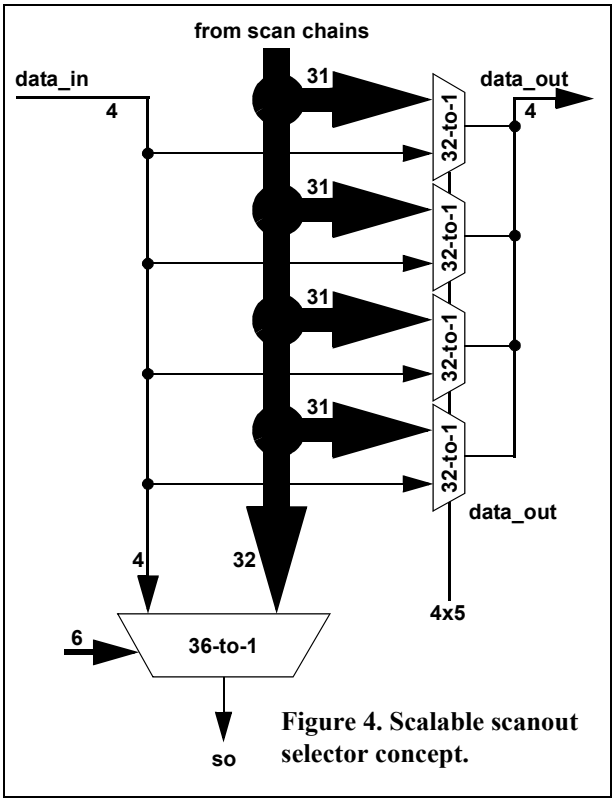

Figure 4. Scalable scanout selector concept.

3.3.1. Proposed internal architecture

The proposed scalable selector architecture (Figure 5) preserves the concept of Figure 4, but requires much fewer gates and control bits; it has a two-level configuration. The first level of the selector consists of 16 4-to-1 MUXes that select between the 16 **data_in** inputs from the previous selector and the 32 **scan chains**. Choosing 16 as the number of data connections between selectors effectively limits max_side_routing to 17, but this is not a practical limitation because propagation delay typically limits max_side_routing to much lower values. The 16 odd-numbered scan chain inputs connect to two MUXes each, so that MUX g, with g = 0, 1,...15, is connected as follows:

- MUX input 0: data_in[g]
- MUX input 1: scan chain[2g] (even)
- MUX input 2: scan chain[2g+1] (odd)
- MUX input 3: scan chain[(2g+3) modulo 32] (odd)

For example, chain 0 connects to MUX 0, and chain 1 connects to MUX 0 and MUX 15. This configuration, in conjunction with the select logic (section 3.3.2) and the routing function (section 3.3.3) guarantees that any two scan chains can be simultaneously observed ("Scalable selector" column in Table 1).

The 16 outputs of the first level are connected to the next selector (**data_out**) and to a second-level 16-to-1 MUX that selects one output for observation at **so**. Note that routing a scan chain through multiple selectors incurs only the delay of a first-level 4-to-1 MUX per selector

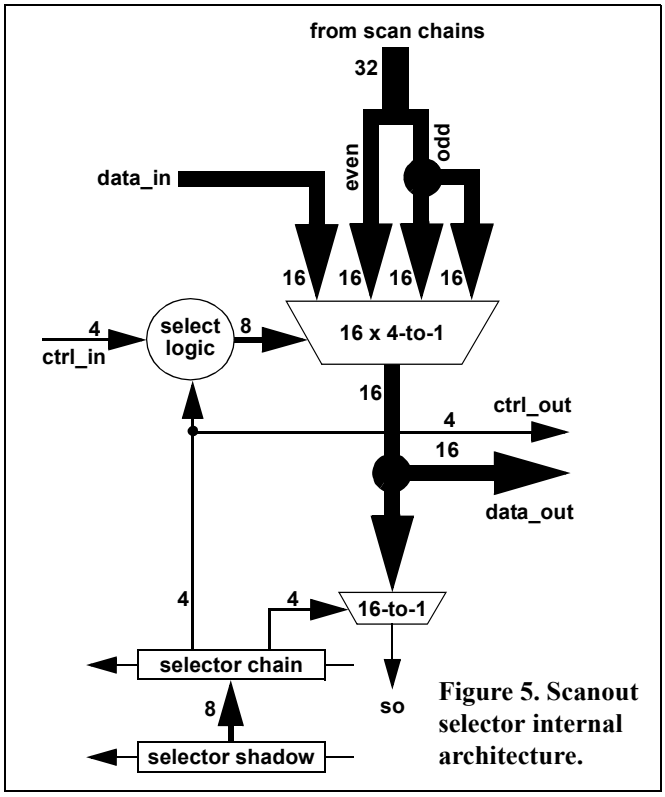

Figure 5. Scanout selector internal architecture.

— a significant improvement over the 32-to-1 MUX in Figure 4. Also, the implementation of the 4-to-1 MUX can be optimized so that the delay from the **data_in** input is minimal.

3.3.2. Select logic

At first, it would appear that the scanout selector requires two control bits for each of the 16 first-level 2-to-1 MUXes and four control bits for the second-level 16-to-1 MUX, or a total of 36 control bits! Such overhead would be unacceptable and, by comparison, the simple selector needs only five control bits and the modular selector [17] only 10 (Table 1). Fortunately, the number of control bits can be greatly reduced through some careful trade-offs.

To minimize the delay from scan chains to outputs, it is preferable to route a scan chain straight down the selector output whenever possible. Therefore, the four select control bits required by the second-level 16-to-1 MUX (Figure 5) are used for every pattern. However, only a few of the 16 first-level 2-to-1 MUXes are active in a pattern: one to select the chain observed straight through the selector, and up to max_side_routing-1 more to select additional chains routed to the next selector. Therefore, the 16 first-level 2-to-1 MUXes are grouped (modulo 4) into four sets, with each set sharing the two control bits, so that only eight bits are needed to control the 16 MUXes. Despite the control limitations imposed by this sharing scheme, any two chains can still be observed by exploiting the multiple chain connections outlined in section 3.3.1. The number of control bits needed is now 8 + 4 = 12, much better than 36, but still not as good as the simple and the modular selectors.

Finally, it can be shown that at most four bits are needed to route any two chains through the first level of the selector, so only four control bits can be used for the first level and eight control bits total (Figure 5). Nonetheless, more than four control bits may be needed to simultaneously observe more than two chains connected to the same

selector. Therefore, the four control bits for the first level of the selector are shared with the first level of the next selector so that each selector can use up to eight bits for the first level (Figure 5). Note that the control bits are shared in the same direction as the data bits; i.e., to the next selector on the right. So, when routing multiple scan chains connected to the same selector, data values are passed onto the next selector while additional control bits are "borrowed" from the previous selector. In effect, $4(d + 1)$ control bits are available to route scan data through d selectors. The select logic (Figure 5) consists of eight XOR gates configured so that control-bit usage is "spread out" over all available bits.

3.3.3. Routing function

In traditional deterministic scan-ATPG, a fault is considered observed as soon as the fault effect has been sensitized and propagated to a primary output or scan cell. In XDBIST, however, only a fraction of the scan cells are observed during a pattern, so special consideration must be given to observable scan chains. Therefore, the test generator was modified to include chain observation as part of the conditions required for fault detection. A primary target fault is first chosen and the scan chain where the fault effect was observed is added to a new chain set. A secondary target fault is then chosen, which can be observed in a chain already in the chain set or in a new chain that can be added to the chain set; a new chain can only be added to the chain set if all chains can be simultaneously routed. If the test generator succeeds in expanding the pattern to test the secondary fault, the new chain, if any, is added to the chain set. Similarly, additional secondary faults are added to the pattern. When no further secondary faults can be found, the pattern is complete and the selector control values are computed and stored as part of the pattern. Next, the fault simulator determines all the chains observed when using the stored selector control data, which may include additional chains not targeted by the test generator.

To determine if a set of scan chains can be observed, even-numbered chains are routed first (Figure 6) because each connects to only a single first-level MUX (section 3.3.1). Next, odd-numbered chains are routed through either of the first-level MUXes. If routing all chains has succeeded, a system of equations is created for the control bits; and, if a solution exists, all selected chains can be observed.

Figure 6 shows the algorithm repeatedly called to route one chain. First, an attempt is made to route the chain straight down through the current selector; if this is not possible, side routing is used to reach the next selector. At each selector, the first-level MUX (or MUXes) the chain connects to is analyzed; and, if possible, control bits are set to route the chain through the MUX. If both first- and second -level routing are successful, then a path has been found to route the chain. Successful completion of the routing depends upon the existence of a solution for the system of equations formed by all control bits that require set values.

3.3.4. Flexible compression ratio

The scalable selector offers an additional dimension of flexibility not found in other selectors — the compression ratio can be adjusted both higher and lower than 32:1. First, note that not all 32 inputs of the selector must be connected to scan chains. Actually, a selector can have *no* inputs connected to scan chains and be used only to enhance the side routing ability for other selectors. For example, consider two selectors with 20 scan chains connected to selector 0 and no scan chains connected to selector 1; the resulting structure has 20 inputs and

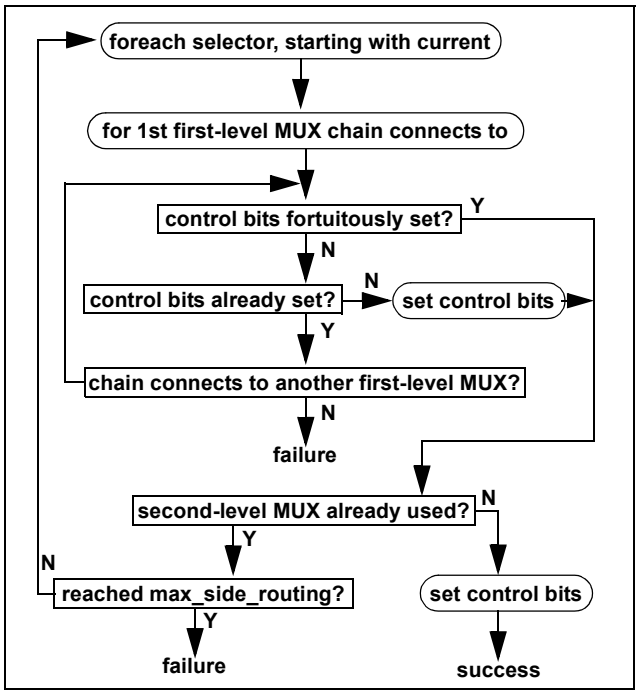

Figure 6. Routing one chain.

two outputs and forms a 10:1 selector. A significant fraction of the gates in the first-level MUXes of selector 1 and some in selector 0 are unused and can be optimized out, reducing the area.

To obtain compression ratios higher than 32:1, not all selectors are connected to an output pin. The function of the selector is unchanged, and only the "second-level MUX already used" step shown in Figure 6 needs to be adjusted to always return "Y" for selectors with no output pin connection. For example, if only half of all selectors have an output connection, the resulting structure forms a 64:1 selector. Theoretically, any compression ratio is possible up to the limits set by max_side_routing. In practice, however, too high a compression may result in an increased number of patterns to achieve the desired test coverage, thus reducing the benefit of higher compression. Evidently, the second-level MUX is unused in selectors without an output pin so a significant number of gates can be optimized out, reducing the area. Alternately, all selectors can be wired to output pins and the compression ratio can be decided when test generation is performed. This offers the unique flexibility of controlling the compression ratio entirely in software. Because unused output pins need not be contacted by the tester, both high and a low pin count testers can be used to test the same package with two different pattern sets, but with the same coverage. Or, the device could be re-used as part of a different system-on-chip (SoC), with a different number of pins.

4. Experimental Results

The input and output components of XDBIST have been implemented as generic library elements that can be automatically instantiated and technology-mapped during test synthesis. The decompressor (Figure 1) offers the choice of different-length PRPG-LFSRs, with matching PRPG shadows and phase shifters. The internal scan chains and the observe selector are automatically configured based on the available number of output pins and the desired compression ratio. All unused gates are eliminated during test-synthesis optimizations, reducing the area overhead of the inserted XDBIST test logic.

Table 1. Selector Analysis (32:1 and 64:1)

	Simple selector	Modular selector [17]	Scalable selector
#chains:	any	max. 512	**any**
compression:	hardwired	32:1	**selectable**
Delay (#MUX):	32:1: 5 64:1: 6	32:1: 7 64:1: na	**6+**
Wiring (fanin/fanout):	32:1: 37/1 64:1: 70/1	32:1: 20/2 64:1: na	**20/1.5**
Control bits / output:	32:1: 5 64:1: 6	32:1: 10 64:1: na	**32:1: 8** **64:1: 12**
Total area ex. 512 chains	32:1: 2688 64:1: 2088		**32:1: 4992** **64:1: 4480**
Total area ex. 128 chains	32:1: 672 64:1: 522	32:1: 5997 64:1: na	**32:1: 1248** **64:1: 1120**
Total area ex. 64 chains	32:1: 336 64:1: 261		**32:1: 624** **64:1: 560**
#chains / #outputs	**Probability to observe all selected chains (32:1)**		
1/*	100.00%	100.00%	**100.00%**
2/2	48.50%	n.a.	**100.00%**
2/5	77.30%	n.a.	**100.00%**
2/10	89.50%	n.a.	**100.00%**
2/16	94.00%	100.00%	**100.00%**
4/5	31.70%	n.a.	**83.70%**
4/10	57.70%	n.a.	**94.70%**
4/16	78.00%	100.00%	**98.10%**
8/10	2.20%	n.a.	**48.10%**
8/16	24.70%	98.70%	**81.50%**

The "Scalable selector" column in Table 1 illustrates a significant characteristic of the implementation — scan-chain-to-output delay for straight-down routing is only six MUXes (two for the first level and four for the second level — Figure 5); the delay increases by two for every additional side routing step. All MUXes are optimized to meet the required delay from scan chain to output pins and may be implemented with NAND/NOR gates. The two-level configuration results in the lowest maximum fanin (20) and lowest average fanout (1.5) of all selectors shown. Total area overhead is only about twice the area of the simple selector, and, for small designs, dramatically lower than the modular selector.

The probabilities of observing the selected set of scan chains are, in some cases, lower than the modular selector (Table 1); but, because checking selection availability is integrated in the test-generation process, there was no pattern-count inflation observed between the modular selector and the scalable selector. Note that the modular selector supports only a 32:1 compression ratio and 512 scan chains; therefore, the comparison between the two selectors is possible only on a small subset of the designs supported by the scalable selector.

The CPU time required by the routing algorithm is but a small part of the total test-generation time. The dominant part of the routing-function CPU time is taken by the equation solver for the selector control values. However, even for a large design (e.g., 512 scan chains connected to a single compressor-decompressor), the selector chain has only 128 bits; thus, the system of equations has 128 variables. The CPU time to solve a system of equations increases with the square of the number of variables. By comparison, the system of equations for the selector discussed in [17] has 160 variables and the system of equations for PRPG seed computation has 479 variables.

5. Conclusions

We have presented a novel scanout compression architecture for X-tolerant deterministic BIST that allows flexible compression ratios, scales to any number of scan chains and minimizes area overhead and control-data size. Compression ratio and scan-chain-to-output delay are software-selectable. The architecture guarantees that any two chains can be observed in any pattern. All observation probabilities are high enough so that the number of patterns does not increase. XDBIST test-coverage, full X-tolerance and scan-based diagnosis ability are preserved and are the same as deterministic scan-ATPG.

References

[1] Semiconductor Industry Association (SIA), *International Technology Roadmap for Semiconductors (ITRS)*, 1999.

[2] M. Abramovici, M.A. Breuer, A.D. Friedman, *Digital Systems Testing and Testable Design*, IEEE Press, 1990.

[3] E.B. Eichelberger, E. Lindbloom, J.A. Waicukauski, T.W. Williams, *Structured Logic Testing*, Prentice-Hall, 1991.

[4] P.H. Bardell, W.H. McAnney, J. Savir, *Built-In Test for VLSI: Pseudorandom Techniques*, John Wiley & Sons, 1987.

[5] B. Könemann, "LFSR-Coded Test Patterns for Scan Designs", *European Test Conference*, Munich, 1991.

[6] P. Wohl, J.A. Waicukauski, S. Patel, M. Amin, "Efficient Compression and Application of Deterministic Patterns in a Logic BIST Architecture", *Design Automation Conf.* 2003, pp. 566-569.

[7] J. Rajski, J. Tyszer, M. Kassab, N. Mukherjee, R. Thompson, K.H. Tsai, A. Hertwig, N. Tamarapalli. G. Mrugalski, G. Eide, J. Qian, "Embedded Deterministic Test for Low Cost Manufacturing Test", *International Test Conf.* 2002, pp. 301-310.

[8] H.J. Nadig, "Testing a Microprocessor Product Using a Signature Analysis", *International Test Conf.* 1978, pp.159-169.

[9] P.H. Bardell, W.H. McAnney, "Self-Testing of Multichip Logic Modules", *International Test Conf.* 1982, pp.200-204.

[10] G. Hetherington, T. Fryars, N. Tamarapalli, M. Kassab, A. Hassan, J. Rajski, "Logic BIST for Large Industrial Designs: Real Issues and Case Studies", *Intern. Test Conf.* 1999, pp.358-367.

[11] C. Barnhart, V. Brunkhorst, F. Distler, O. Farnsworth, B. Keller, B. Koenemann, "OPMISR: The Foundation for Compressed ATPG Vectors", *International Test Conf.* 2001, pp. 748-757.

[12] I. Pomeranz, S. Kundu, S.M. Reddy, "On Output Response Compression in the Presence of Unknown Output Values", *Design Automation Conference* 2002, pp. 255-258.

[13] M. Naruse, I. Pomeranz, S.M. Reddy, S. Kundu, "On-chip Compression of Output Responses with Unknown Values Using LFSR Reseeding", *International Test Conference* 2003.

[14] S. Mitra, K.S. Kim, "X-Compact An Efficient Response Compaction Technique for Test Cost Reduction", *International Test Conference* 2002, pp.311-320.

[15] J. Rajski, J. Tyszer, C. Wang, S.M. Reddy, "Convolutional Compaction of Test Responses", *International Test Conference* 2003.

[16] P. Wohl, J.A. Waicukauski, S. Patel, G. Maston, "Effective Diagnostics through Interval Unloads in a BIST Environment", *Design Automation Conference* 2002, pp. 249-254.

[17] P. Wohl, J.A. Waicukauski, S. Patel, M. Amin, "X-Tolerant Compression and Application of Scan-ATPG Patterns in a BIST Architecture", *International Test Conference* 2003.