Open access • Proceedings Article • DOI:10.1109/VTS.2001.923409

Hybrid BIST based on weighted pseudo-random testing: a new test resource partitioning scheme — Source link

A. Jas, C.V. Krishna, Nur A. Touba

Institutions: University of Texas at Austin

Published on: 29 Mar 2001 - VLSI Test Symposium

Topics: Fault coverage, Automatic test pattern generation, Overhead (computing) and Built-in self-test

Related papers:

- Reducing test data volume using external/LBIST hybrid test patterns
- LFSR-coded test patterns for scan designs
- Bit-flipping BIST
- Test vector encoding using partial LFSR reseeding
- Embedded deterministic test for low cost manufacturing test

Share this paper:

View more about this paper here: https://typeset.io/papers/hybrid-bist-based-on-weighted-pseudo-random-testing-a-new-25w369b89m

Hybrid BIST Based on Weighted Pseudo-Random Testing: A New Test Resource Partitioning Scheme

Abhijit Jas, C.V. Krishna, and Nur A. Touba

Computer Engineering Research Center
Department of Electrical and Computer Engineering
University of Texas, Austin TX 78712-1084
E-mail: {jas, krishna, touba}@ece.utexas.edu

Abstract

This paper presents a new test resource partitioning scheme that is a hybrid approach between external testing and BIST. It reduces tester storage requirements and tester bandwidth requirements by orders of magnitude compared to conventional external testing, but requires much less area overhead than a full BIST implementation providing the same fault coverage. The proposed approach is based on weighted pseudo-random testing and uses a novel approach for compressing and storing the weight sets. Three levels of compression are used to greatly reduce test costs. No test points or any modifications are made to the function logic. The proposed scheme requires adding only a small amount of additional hardware to the STUMPS architecture. Experimental results comparing the proposed approach with other approaches are presented.

1. Introduction

The continual increase in integration density for VLSI has made system-on-a-chip (SOC) designs possible. The amount of test data volume required for testing such large designs is growing rapidly. Conventional external testing approaches where all test data is stored on the tester and transferred to/from the circuit-under-test (CUT) is becoming increasingly difficult. Testers have limited speed, memory, and I/O channels. The limited test data bandwidth (see Fig. 1) between the tester and the chip is becoming a major bottleneck that is expected to become much worse as the projections in [Khoche 00] indicate. There is a need for new test resource partitioning schemes that reduce test data bandwidth requirements and reduce tester storage requirements by orders of magnitude.

One well-known approach is to use built-in self-test (BIST). BIST involves performing test pattern generation and output response compaction on the chip. BIST has been studied for many years. The most economical BIST schemes are based on pseudo-random pattern testing. The problem with pseudo-random pattern testing, however, is that it generally does not provide high enough fault coverage due to the presence of random-pattern-resistant faults [Eichelberger 83]. There are two solutions to this problem. One is to modify the circuit to eliminate the random pattern resistance by inserting test points [Eichelberger 83], and the other is to modify the test pattern generator by adding additional hardware to generate patterns that detect the hard faults [Touba 96], [Kiefer 98, 00], [Fagot 98]. Both approaches have significant drawbacks. Test point insertion requires modifying the function logic which can degrade system performance, and modifying the test pattern generator can require large amounts of additional silicon area.

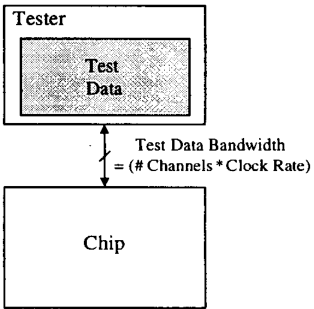

Figure 1. Block Diagram Illustrating Test Data Bandwidth

In this paper, we present a new test resource partitioning scheme that is a hybrid approach between BIST and external testing. The term "hybrid BIST" will be used in this paper to classify any scheme that involves combining external data from the tester along with BIST hardware on the chip to provide a hybrid test solution for a particular module or core. A hybrid BIST approach reduces the test data stored on the tester compared with full external testing, but it does not require as much hardware overhead as full BIST. There are several existing approaches that can be classified as hybrid BIST approaches. A simple approach for hybrid BIST is to use a STUMPS architecture [Bardell 82] to apply pseudo-

1093-0167/01 $10.00 © 2001 IEEE

random patterns to detect the random pattern testable faults, and then use deterministic scan vectors from the tester to detect the hard faults. There have been two recent case studies on using this approach for large industrial designs [Hetherington 99], [Pressly 99]. The case study in [Pressly 99] was done on the Motorola PowerPC™ microprocessor core, and the study in [Hetherington 99] was done on large ASIC designs. In [Pressly 99], the reduction in external test storage requirements after using 500K BIST patterns was around 30%. In [Hetherington 99], test points were inserted, but the reduction in test storage requirements after 262K BIST patterns still ranged only from 35% to 55%. What these results indicate is that most of the vectors in a deterministic test set target hard faults which are missed by BIST. So a straightforward hybrid BIST approach where pseudo-random vectors are applied with BIST hardware followed by deterministic vectors from the external tester, can only achieve a limited reduction in tester storage requirements, generally not an order of magnitude reduction.

In [Das 00], a hybrid BIST approach was proposed where some of the scan chains in a STUMPS architecture are filled with deterministic test data from the tester while the rest of the scan chains are filled from the pseudo-random pattern generator (PRPG). The set of scan chains receiving deterministic data is rotated in a round-robin fashion. This approach was applied to the Motorola PowerPC™ microprocessor core. Results indicated that the test storage requirements could be reduced by around 50% with this approach compared with 31% as was reported in [Pressly 99] for using fully pseudo-random patterns followed by fully deterministic patterns.

In this paper, we propose a new hybrid BIST approach that is based on weighted pseudo-random testing. Weighted pseudo-random testing involves biasing the generation of pseudo-random patterns towards those that detect the hard faults. A "weight" is assigned to each bit position in a test vector and corresponds to the probability of a '1' being generated at that bit position. Because of conflicting requirements for detecting hard faults in a circuit, multiple weight sets are generally required [Wunderlich 88]. Some number of weighted pseudo-random patterns are generated for each weight set to detect all of the faults. There are two types of weighted pseudo-random testing schemes, one for external testing and one for BIST. For external testing, the weight sets are stored in the tester memory, and the weighted pseudo-random pattern generation is performed on the tester as each test vector is being transferred to the chip (as illustrated in Fig. 2) [Waicukauksi 89]. This approach reduces tester memory requirements, but it does not help with the test data bandwidth bottleneck problem because all of the test data still has to be transferred from the tester to the chip. The other scheme for weighted pseudo-random testing is to use it for BIST (as illustrated in Fig. 3). In this case, the weight sets are stored on the chip, and on-chip hardware is used to generate the weighted pseudo-random patterns [Brglez 89], [Muradali 90], [Pomeranz 93a] (or the hardware could be placed on a separate "test chip" [Ströle 91]). The problem with a full BIST implementation of weighted pseudo-random testing is that storing the weight sets on the chip requires an enormous amount of area overhead.

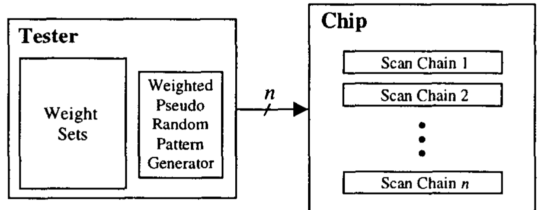

Figure 2. Weighted Pseudo-Random Pattern Generation for External Testing

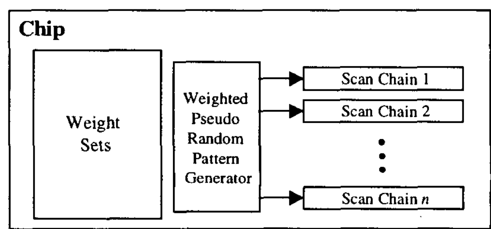

Figure 3. Weighted Pseudo-Random Pattern Generation for BIST

In this paper, we propose a novel hybrid weighted pseudo-random scheme that reduces tester storage requirements and solves the test data bandwidth bottleneck problem, but does not require the area overhead of a full BIST implementation. It uses three levels of compression to provide orders of magnitude reduction in tester storage requirements. In comparison with the approach of performing the weighted pattern generation on the tester, the proposed approach not only reduces tester memory requirements, but more importantly, it also reduces the test data bandwidth requirements from the tester to the chip. If weighted pattern generation is performed on the tester and then used to drive 32 scan chains, it requires 32 channels from the tester, whereas the proposed approach can drive the same number of scan chains with data coming from only a small number of channels from the tester. As system-on-a-chip designs become larger and more complex, this capability will be essential to keep test time down. Note that test time is lower bounded by the total amount of test

data stored on the tester divided by the test data bandwidth between the tester and chip (which is limited by the number of I/O pins on the chip and I/O channels from the tester).

A simple approach for implementing a hybrid BIST weighted pseudo-random scheme would be to store all the weight sets on the tester, and then transfer one weight set at a time to the chip. After some number of weighted pseudo-random vectors are generated on the chip for one weight set, the next weight set could be transferred from the tester to the chip. The problem with this approach is that at least 2 bits (or more depending on the precision of the weights) are needed to encode the weight value for each scan element in a design. This means that the storage requirements on the chip for one weight set would be at least double the number of scan elements in the design which would be an enormous area overhead. Fortunately, it turns out that weight sets are highly compressible. This fact is greatly exploited in the scheme proposed in this paper. We present a novel hybrid BIST weighted pseudo-random testing scheme that uses only a small amount of data from the tester to significantly reduce BIST hardware requirements on the chip. The proposed approach reduces tester storage requirements by orders of magnitude compared to full external testing while requiring much less overhead than a full BIST approach that provides the same fault coverage. No test points or any modifications are made to the function logic. The proposed scheme requires adding only a small amount of additional hardware to the STUMPS architecture.

2. Overview of the Proposed Scheme

This section describes the basic idea of the proposed scheme for hybrid BIST with weighted pseudo-random testing. The implementation details are explained later in subsequent sections of the paper. Figure 4 shows a block diagram of the test architecture. In this scheme, 3-valued weights are used (as was proposed in [Pomeranz 93a]), i.e., the three possible weights for a specific scan element are **0**, **1**, and **u** (which signifies "unbiased"). A weight of **0** forces the value of a particular scan element to *0*, a weight of **1** forces it to *1*, and a **u** means that the scan element takes on a value of *0* or *1* with equal probability. In Fig. 4, the scan elements of the chip have been configured into n scan chains each of which contains m scan elements (bits). Since a 3-valued weight system is being used, two bits are required to store the weight for each scan element. The encoding used in this particular example for the three weights are $(w1_i, w0_i) = 01$ for weight **0**, *10* for weight **1**, and *00* for **u**. However any other 3-valued encoding can be used.

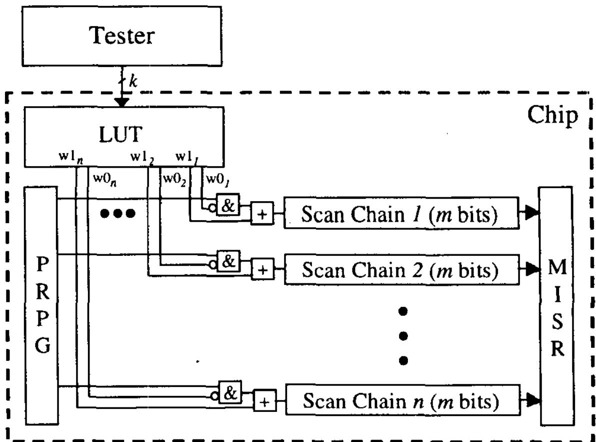

Figure 4. Block Diagram of Proposed Hybrid BIST Test Architecture

Figure 4 illustrates the STUMPS architecture for BIST where at each clock cycle, n pseudo-random bits generated by the pseudo-random pattern generator (PRPG) are scanned into the n scan chains (one bit into each scan chain). However, in the proposed approach the pseudo-random bits are transformed by the logic at the input of the scan chains according to the weight bits $w1_i$, $w0_i$. The weight bits are stored in a look-up table (LUT) on the chip. Details of the LUT will be explained in the next section. At each clock cycle, the set of weights corresponding to the i-th bit of every scan chain is looked up from the LUT and used to transform the pseudo-random bits coming out of the PRPG to generate the weighted pseudo-random bits which are then scanned into the scan chains. It takes m scan clock cycles to completely fill the n scan chains. Once the scan chains are filled (i.e., nm scan bits have been shifted into the scan chains) the system clock is applied and the output response is captured in the scan chains. This output response is shifted out and compacted in the multiple input signature register (MISR) as the next test vector is shifted in.

The $2n$ weight bits for the n scan chains $(w1_1,w0_1)$, $(w1_2,w0_2)$, ... $(w1_n,w0_n)$ are stored in one location of the LUT. At each clock cycle, the tester supplies an LUT index which is used to read the weights for the n bits from the LUT. These weights are then used to transform the n pseudo-random bits coming from the PRPG as they are shifted into the scan chains. The tester and the PRPG operate at the same clock frequency in a lock-step manner. The number of bits required for the LUT index depends on the size of the LUT. The number of bits, k, required for the index is generally much less than n. So in this scheme, k tester channels are being used to drive n

scan chains, where k is much less than n. Hence, the tester bandwidth requirements are being reduced.

For each weight set, a sequence of m LUT indices are stored on the tester. If L weighted pseudo-random patterns are to be generated for each weight set, then the tester simply resends the sequence of indices for each weight set L times. Only one copy of the sequence of indices for each weight set needs to be stored in the tester memory.

There are three levels of compression in this scheme. The first level of compression is that only the unique parts of each weight set need to be stored in the LUT (this will be explained in detail in the next section), thus for each weight set there will much less than m rows in the LUT. The second level of compression is that each weight set is stored as a sequence of k-bit indices on the tester where k scales logarithmically with the number of rows in the LUT and is much less than n. The third level of compression is that each weight set is expanded into L weighted pseudo-random test patterns. These three levels of compression result in greatly reduced tester storage requirements and tester bandwidth requirements.

3. Determining Contents of LUT

The first step is to determine the weight sets that will be required to achieve the desired fault coverage. Any number of weighted pseudo-random patterns can be generated for each weight set with the scheme presented here. Many techniques for determining weight sets for a particular CUT have been proposed in the literature, e.g., [Waicukauski 89], [Pomeranz 93a], [Bershteyn 93]. Any of these techniques can be used.

Given the weight sets, the next step is to determine the contents of the LUT. Figure 5 illustrates how the weight sets are stored in the LUT and accessed during the testing. Let n denote the number of scan chains and m denote the number of bits in each scan chain. Thus the total number of scan elements is nm. As mentioned earlier, since a three-valued weight system is being used, two bits are required to store the weights for each scan element. The weights for the i-th scan element of all the scan chains are stored in each location of the LUT so that they can all be read in the same clock cycle and used to transform the pseudo-random bits coming in from the PRPG. Thus, $(w1_1,w0_1)$, $(w1_2,w0_2)$, ... $(w1_n,w0_n)$, denotes the weights represented by two bits for the i-th scan element in each of the n scan chains. The rows r_{11}, r_{12}, ... r_{1m} correspond to the first weight set. There is one row for each of the m bits of the scan chains. r_{21}, r_{22}, ... r_{2m} correspond to the second weight set.

The LUT can be compressed to a great extent by merging identical rows as is illustrated in Fig. 5. This results in a lot of compression because many of the weight sets will have similar assignments in various rows in the LUT. Very often, the i-th scan elements of the n scan chains will all be assigned **u**. This common case reduces to a single row in the LUT. A small example showing how two weight sets are compressed in the LUT is shown in Fig. 6. Reducing the size of the LUT has a two-fold advantage. Not only does it reduce hardware requirements, but it also reduces the size of the indices as fewer bits are now required to index the LUT.

Once the LUT has been constructed, then each weight set can be stored as a sequence of m indices on the tester where each index is $\lceil log_2 p \rceil$ bits wide where p is the total number of rows in the LUT.

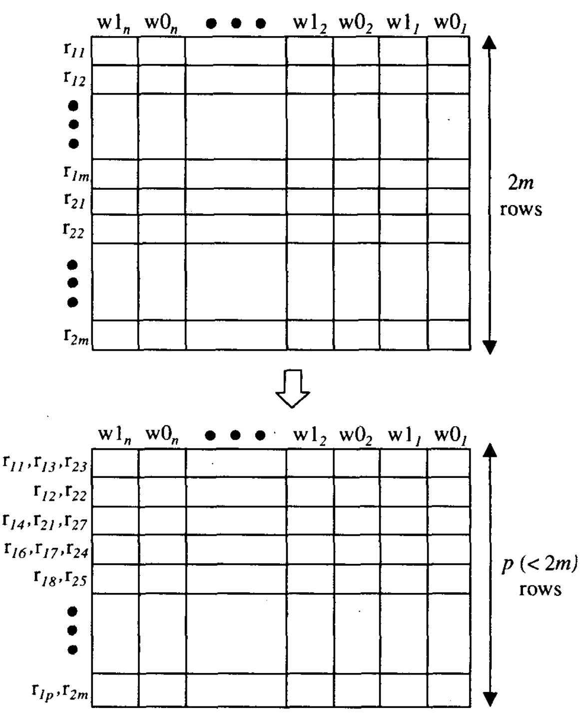

Figure 5. Storage of Weight Sets in LUT

A small example is shown in Fig. 6. The chip-under-test has 4 scan chains each of which is 10 bits long. Assume that 5 weighted pseudo-random patterns will be generated for each weight set for a total of 10 weighted pseudo-random patterns as shown in Fig. 7. In Fig. 7, the bold columns in the figure show the bits that are fixed because of the weight sets. The **u** bits take on a value of *1* or *0* randomly.

Fig. 6 shows how the weights will be stored in the LUT on the chip. Weights for a certain bit position for all the 4 scan chains will be stored in one location of the

LUT. Thus every location of the LUT will be 8 bits wide (2 bits to represent each weight) and there will initially be 10 LUT locations for each weight set for the 10 bit positions in the scan chains. The scan chains are denoted by *s1*, *s2*, *s3*, and *s4*, and the bit positions are denoted by Bit *i*. Since the weight sets have a lot of similarities, they can be compressed to a great extent. In Fig. 6, the unique weight patterns are shown in bold. Thus, the duplicate patterns can be eliminated and only the unique patterns stored in the LUT. Hence the LUT storage requirements can be reduced from 20 rows to only 8. Only a sequence of *m* LUT indices needs to be stored in the tester for each weight set. In this case, each index is only 3 bits wide since there are only 8 rows in the LUT.

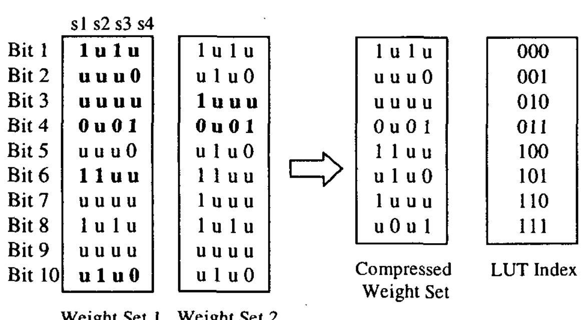

Figure 6. Small Example Illustrating Proposed Approach

	scan chain 1	scan chain 2	scan chain 3	scan chain 4
weight 1	1uu0u1u1uu	uuuuu1uuu1	1uu0uuu1uu	u0u10uuuu0
weight 2	1u10u111uu	u1uu11uuu1	1uu0uuu1uu	u0u10uuuu0

	scan chain 1	scan chain 2	scan chain 3	scan chain 4
weighted random pattern set 1	1010110100	0110010101	1010110100	1011010100
	1100111110	0111011011	1010001110	1001000010
	1100010111	1010110001	1100111111	1001001010
	1000111101	1101111111	1110010100	0001011100
	1010011110	0110111101	1000101110	0011011000
weighted random pattern set 2	1010111100	1110111011	1000001110	10010011u0
	1010111101	0101110101	1010100110	1001000010
	1110011110	1101111001	1100110101	1001011000
	1110011111	1101110001	1010101101	0011001100
	1010011111	0101110101	1100011100	0011011100

Figure 7. Example of Weighted Pseudo-Random Patterns Generated for 2 Weight Sets

4. Hardware Implementation of LUT

The LUT can be implemented in hardware in a variety of ways. One simple way of implementing the LUT is by using a RAM. Generally there are a lot of RAM's already present in a design, so it may be possible to use one of them to serve as the LUT for this scheme. In this case, the contents of the LUT would be initially stored in the tester. Before the testing begins, the tester would initialize the RAM with the proper contents. The minimum size required for the RAM would depend on the number of rows in the LUT and the number of scan chains. Note that if the RAM is larger than necessary, this does not present a problem. In such a case, only a subset of the addressable locations in the RAM would be used, and only a subset of the data bits stored at each address would be used.

Another way of implementing the LUT would be to use a PLA (programmable logic array). Because most of the weight values are **u** (which could be encoded using one specified bit and one don't care), the number of rows in the PLA can be greatly minimized. A PLA provides a very compact and efficient implementation of the LUT.

5. Experimental Results

Experiments were performed on the 5 largest ISCAS-89 circuits. For each circuit, STUMPS architectures with different numbers of scan chains were constructed. First, 32,000 pseudo-random patterns were applied with the STUMPS architecture to detect the random pattern testable faults. Then the remaining random pattern resistant faults were targeted using weighted pseudo-random patterns based on the 3-valued weight system described in Sec. 2. The weight sets were selected using the procedure described in [Pomeranz 93a] with 1,000 weighted pseudo-random patterns being generated for each weight set. Table 1 shows the number of weight sets that were required for each circuit to achieve 100% coverage of detectable faults. For each circuit, Table 1 shows the total number of scan elements and the results for dividing them into different numbers of scan chains. In each case, the hardware requirements are shown for using either a RAM or a PLA to implement the LUT. The amount of test data that must be stored on the tester is shown for the proposed hybrid BIST approach and for conventional external testing using the highly compacted test vectors obtained with COMPACTEST [Pomeranz 93b]. The compression ratio for the test data storage requirements is shown. It is computed as:

(test data for conventional external testing) / (test data for proposed hybrid BIST approach)

As can be seen, the tester storage requirements are reduced by orders of magnitude with the proposed hybrid BIST approach.

Table 2 shows a comparison of the proposed hybrid BIST approach versus an approach where BIST is used to detect the random pattern testable faults and then "top-up" deterministic test vectors are applied from the tester to detect the random pattern resistant faults. The "top-

up" deterministic vectors were obtained by first applying 32,000 pseudo-random patterns using the STUMP architecture, and then doing ATPG for the remaining undetected faults. The compression ratio for the test data storage requirements is shown. It is computed as:

(test data for "top-up" test vectors) / (test data for proposed hybrid BIST approach)

As can be seen, the tester storage requirements are reduced by at least an order of magnitude in all cases with the proposed hybrid BIST approach based on weighted pseudo-random testing.

Table 3 shows a comparison of the area overhead for the proposed hybrid BIST approach compared with the best published results for deterministic BIST in [Kiefer 98]

Table 1. Comparison of Proposed Hybrid BIST Scheme with External Testing

Circuit		Num.	Num.	RAM	PLA			Test Data		Test Data
Name	Scan Elements	Weight Sets	Scan Chains	Size (Bytes)	Inputs	Outputs	Rows	Hybrid BIST	External Testing	Compression Ratio
s9234	247	8	8	240	7	16	96	1729	66690	38
			16	452	7	32	99	865		77
			32	512	6	64	62	371		180
S13207	700	1	8	84	6	16	35	525	331800	632
			16	168	6	32	35	263		1261
			32	176	5	64	22	110		3016
S15850	611	4	8	198	7	16	74	2139	150306	70
			16	436	7	32	100	1070		140
			32	608	7	64	75	535		281
S38417	1664	10	8	642	9	16	255	18720	316160	17
			16	1580	9	32	358	9360		34
			32	2336	9	64	271	4680		67
			50	3100	8	100	227	2663		119
S38584	1464	3	8	114	6	16	44	3294	366000	111
			16	340	7	32	64	1922		190
			32	712	7	64	65	961		381
			50	887	7	100	53	615		595

Table 2. Comparison of Proposed Hybrid BIST Scheme with BIST Followed by Top-Up Test Patterns from Tester

Circuit		Num.	Test Data		Test Data
Name	Scan Elements	Scan Chains	Hybrid BIST	BIST + Top-Up	Compression Ratio
S9234	247	16	865	32544	37
s13207	700	16	263	26600	101
s15850	611	16	1070	50102	47
s38417	1664	32	4680	158080	34
s38584	1464	32	961	61488	64

Table 3. Comparison of Proposed Hybrid BIST Scheme with Deterministic BIST Scheme Described in [Kiefer 98]

Circuit		Hybrid BIST PLA				Deterministic BIST PLA				Area Overhead
Name	Scan Elements	Inputs	Outputs	Rows	Area	Inputs	Outputs	Rows	Area	Compression Ratio
S9234	247	7	16	96	0.8627	14	10	187	1.6023	1.8
s13207	700	6	16	35	0.4042	18	18	56	0.8698	2.1
s15850	611	7	16	74	0.7009	14	22	199	2.4907	3.5
s38417	1664	9	16	255	2.2061	34	68	509	16.4996	7.5
s38584	1464	6	16	44	0.4606	14	46	132	2.8289	6.1

that provides the same fault coverage. This comparison assumes that a PLA implementation is used. It should be noted that for one or both of the techniques, a multilevel logic implementation may be more efficient than a PLA (this is in fact suggested in [Kiefer 98] for a STUMPS architecture). Also, note that the comparison only considers the area overhead of the PLA. It does not include the area overhead for the STUMPS architecture itself, however, the results in [Kiefer 98] indicate that the PLA area dominates the total area for BIST for these circuits. The compression ratio for the PLA area overhead requirements is shown. It is computed as:

(PLA area for [Kiefer 98]) / (PLA area for proposed hybrid BIST approach)

As can be seen, the area overhead is greatly reduced with the proposed hybrid BIST approach. Note that the reduction becomes more pronounced for the larger circuits. For *s38417* and *s38584*, the hardware overhead is reduced by a factor of 7.5 and 6.1, respectively. Of course, it should be noted that the full BIST implementation in [Kiefer 98] does not have any tester storage requirements.

6. Conclusions

The new test resource partitioning scheme presented here combines BIST hardware with external data from the tester to provide a hybrid BIST solution. By using three levels of compression, the tester storage requirements are reduced by orders of magnitude compared to conventional external testing. Compared with using a deterministic BIST scheme to achieve the same fault coverage, it was shown that the area overhead on the chip can be significantly reduced. This new test resource partitioning scheme provides another design point in addition to external testing or deterministic BIST that may be attractive in some test resource partitioning scenarios. Note that in addition to the benefits of reducing tester storage and bandwidth requirements, the proposed approach also provides the benefits of weighted pseudo-random pattern testing in detecting non-modeled defects.

Acknowledgements

This material is based on work supported in part by the National Science Foundation under Grant No. MIP-9702236 and in part by the Texas Advanced Technology Program under Grant No. 003658-0644-1999.

References

[Bardell 82] Bardell, P.H., and W.H. McAnney "Self-Testing of Multichip Logic Modules," *Proc. of International Test Conference*, pp. 200-204, 1982.

[Brglez 89] Brglez, F., G. Gloster, and G. Kedem, "Hardware-Based Weighted Random Pattern Generation for Boundary Scan," *Proc. of International Test Conference*, pp. 264-274, 1989.

[Bershteyn 93] Bershteyn, M., "Calculation of Multiple Sets of Weights for Weighted Random Testing," *Proc. of International Test Conference*, pp. 1031-1040, 1993.

[Das 00] Das, D., and N.A. Touba, "Reducing Test Data Volume Using External/LBIST Hybrid Test Patterns," *Proc. of International Test Conference*, pp. 115-122, 2000.

[Eichelberger 83] Eichelberger, E.B., and E. Lindbloom, "Random-Pattern Coverage Enhancement and Diagnosis for LSSD Logic Self-Test," *IBM Journal of Research & Development*, Vol. 27, No. 3, pp. 265-272, May 1983.

[Fagot 98] Fagot, C., P. Girard, and C. Landrault, "On Using Machine Learning for Logic BIST," *Proc. of International Test Conference*, pp. 338-346, 1998.

[Hetherington 99] Hetherington, G., T. Fryars, N. Tamarapalli, M. Kassab, A. Hassan and J. Rajski, "Logic BIST for Large Industrial Designs: Real Issues and Case Studies," *Proc. of International Test Conference*, pp. 358-367, Sept. 1999.

[Khoche 00] Khoche, A., and J. Rivoir, "I/O Bandwidth Bottleneck for Test: Is it Real?," *Proc. of International Workshop on Test Resource Partitioning*, 2000.

[Kiefer 98] Kiefer, G., and H.-J. Wunderlich, "Deterministic BIST with Multiple Scan Chains," *Proc. of International Test Conference*, pp. 1057-1064, 1998.

[Kiefer 00] Kiefer, G., H. Vranken, E.J. Marinissen, and H.-J. Wunderlich, "Application of Deterministic Logic BIST on Industrial Circuits," *Proc. of International Test Conference*, pp. 105-114, 2000.

[Muradali 90] Muradali, F., V.K. Agarwal, and B. Nadeau-Dostie, "A New Procedure for Weighted Random Built-In Self-Test," *Proc. of International Test Conference*, pp. 660-668, 1990.

[Pomeranz 93a] Pomeranz, I., and S.M. Reddy, "3-Weight Pseudo-Random Test Generation Based on a Deterministic Test Set for Combinational and Sequential Circuits," *IEEE Transactions on Computer-Aided Design*, Vol. 12, No. 7, pp. 1050-1058, Jul. 1993.

[Pomeranz 93b] Pomeranz, I., L.N. Reddy, and S.M. Reddy, "COMPACTEST: A Method to Generate Compact Test Sets for Combinational Circuits," *IEEE Transactions on Computer-Aided Design*, Vol. 12, No. 7, pp. 1040-1049, Jul. 1993.

[Pressly 99] Pressly, M., D. Das, and C. Hunter, "LBIST for PowerPC™ Embedded Core Microprocessors: Feasible or Not?," *International Workshop on Microprocessor Test and Verification*, 1999.

[Ströle 91] Ströle, A.P., and H.-J. Wunderlich, "TESTCHIP: A Chip for Weighted Random Pattern Generation, Evaluation, and Test Control", *IEEE Journal of Solid-State Circuits*, Vol. 26, No. 7, pp. 1056-1063, Jul. 1991

[Touba 96] Touba, N.A., and E.J. McCluskey, "Altering a Pseudo-Random Sequence of Bits for Scan-Based BIST", *Proc. of International Test Conference*, pp. 167-175, 1996.

[Waicukauski 89] Waicukauski, J.A., E. Lindbloom, E.B. Eichelberger, and P. Forlenza, "A Method for Generating Weighted Random Test Patterns," IBM Journal of Research and Development, Vol. 33, No. 2, pp. 149-161, Mar. 1989.

[Wunderlich 88] Wunderlich, H.-J., "Multiple Distributions for Biased Random Test Patterns," *Proc. of International Test Conference*, pp. 236-244, 1988.