REDUCING TEST DATA VOLUME USING EXTERNAL/LBIST HYBRID TEST PATTERNS

Debaleena Das[1,2] and Nur A. Touba[1]

[1]*Computer Engineering Research Center*
Department of Electrical and Computer Engineering
University of Texas, Austin, TX 78712
E-mail: {ddas, touba}@cat.ece.utexas.edu

[2]*Motorola Inc.*
Cores and System Technology
7700 West Parmer Lane
Austin, TX 78729

Abstract

A common approach for large industrial designs is to use logic built-in self-test (LBIST) followed by test data from an external tester. Because the fault coverage with LBIST alone is not sufficient, there is a need to top-up the fault coverage with additional deterministic test patterns from an external tester. This paper proposes a technique of combining LBIST and deterministic ATPG to form "hybrid test patterns" which merge pseudo-random and deterministic test data. Experiments have been done on the Motorola PowerPC^{TM} microprocessor core to study the proposed hybrid test patterns. Hybrid test patterns provide several advantages: 1) can be applied using STUMPS architecture [Bardell 82] with a minor modification, 2) significantly reduce external test data stored in tester memory, 3) reduce the number of pseudo-random patterns by orders of magnitude, thus addressing power issues.

1. Introduction

Logic built-in self-test (LBIST) is being increasingly used to tackle the test problems associated with system-on-a-chip (SOC) and other large industrial designs, the primary being test data volume. LBIST is becoming increasingly important for reducing test data volume in manufacturing test [Hetherington 99], [Pressly 99]. The case study in [Hetherington 99] was done on large ASIC designs and the study in [Pressly 99] was done on the Motorola PowerPC^{TM} microprocessor core. A million gate integrated circuit can require test data in the order of gigabytes. The problem is compounded for SOC designs where the embedded cores have to share tester memory with system logic test patterns. Difficulty in accessing the large number of core inputs and outputs makes functional test an unattractive proposition for core test. Hence, in addition to stuck-at scan test patterns, delay fault scan test patterns are needed to ensure test quality.

Tester memory capacities are generally not sufficient to store the entire stuck-at and path delay scan test patterns for large circuits. Reloads become necessary. Reloads are slow and greatly increase the cost of using the tester. The manufacturing test cost is further increased by the bottleneck in the scan interface between the tester and the circuit-under-test caused by the upper limit on scan frequencies. For these reasons, the test cost per transistor is going up whereas the design cost per transistor is going down [Bottoms 98]. The use of logic BIST can greatly reduce the manufacturing test cost by reducing the volume of stuck-at scan test patterns to be applied by the tester.

Unlike memory BIST, which has been around for some time, logic BIST is relatively new on the industrial scene. The logic BIST architecture most commonly used is based on the STUMPS architecture [Bardell 87], which assumes the design to be a full scan or a partial scan design. Commercially available ATPG software can simulate STUMPS-based LBIST test patterns. The fault coverage achieved by LBIST on large industrial designs can vary between 65 to 80% since the test patterns applied are pseudo-random.

Several publications deal with techniques to increase the fault coverage for LBIST. These techniques can be roughly divided into techniques that modify the circuit-under-test and techniques that deal with the pseudo-random test pattern generator (PRPG).

One of the prominent techniques for modifying the circuit-under-test to increase fault coverage of LBIST test patterns is the insertion of test points [Eichelberger 83], [Touba 96], [Tamarapalli 96]. The unacceptably low fault coverage achieved by LBIST test patterns is due to the inherent random-pattern resistance of large designs [Eichelberger 83]. Test points increase controllability and observability thus making the design more random-pattern testable. The undesirable aspect of test point insertion is that it adds delay along functional paths. This will be especially unacceptable in the scenario where logic BIST is used to test high performance designs.

Techniques that target the pseudo-random test pattern generator are the use of weighted pseudo-random sequences [Muradali 90], using multiple-polynomial linear feedback shift registers with reseeding [Venkataraman 93], [Hellebrand 95], and STARBIST [Tsai 97]. All these approaches assume complete on-chip test, meaning the LBIST test patterns are the only test patterns applied. The assumption of complete on-chip test in previous approaches led to large area overheads and/or increased delay. However, to combat the increasing problem of exploding test data volume, the trend in industry today is to use LBIST to achieve a certain fault coverage and then top-up the coverage using external test patterns.

There have been two recent studies on LBIST for large industrial designs [Pressly 99], [Hetherington 99]. The case study in [Pressly 99] was done on the Motorola PowerPC microprocessor core. The fault coverage obtained using 500K LBIST patterns is around 80%. The case studies in [Hetherington 99] were done jointly by Texas Instruments and Mentor Graphics on four large ASIC designs. Test point insertion as proposed in [Tamarapalli 96] was successfully used to increase fault coverage to around 95% for ASIC designs in [Hetherington 99]. Though test point insertion does increase the fault coverage, 95% may still be unacceptable. To increase the fault coverage beyond what is achieved by LBIST, both studies followed application of LBIST patterns by deterministic test patterns from an external tester.

The steps in this approach of following LBIST by deterministic ATPG are:

1. Run LBIST test patterns for a fixed number of cycles. The number of cycles is determined by the amount of test time to be allocated for LBIST operation. The number of cycles can be calculated given the length of the longest scan chain and the frequency of scan shift. Running LBIST for a fixed number of cycles achieves a certain fault coverage.

2. Do deterministic ATPG to detect the remaining faults to achieve the required fault coverage. The extra deterministic vectors required to top up the fault coverage achieved by LBIST are referred to as the "top-up test patterns" in [Hetherington 99]. We will follow the same terminology in this paper.

The reduction in stuck-at test pattern data volume achieved by these two steps is reported to be around 30% (applying 500K LBIST patterns, with a gate count of around 260K and without test point insertion) in [Pressly 99] and around 35-55% (applying 262K LBIST patterns, with a gate count ranging from 350K to 750K and with test point insertion) in [Hetherington 99].

The current approach of applying LBIST test patterns followed by external patterns suffers from the following two major drawbacks. The reduction in test pattern data may not be significant enough. Even after test point insertion, the reported reduction was around 50% in [Hetherington 99]. Another drawback is the large number of LBIST test patterns used. The problem with a large number of LBIST patterns has more to do with power dissipation than LBIST test time. With increasing clock frequencies, the time to apply even a million LBIST test patterns will become very low. However, it will cause a lot of power dissipation, as each test pattern requires a large number of shift operations and pseudo-random patterns cause excessive circuit switching [Gerstendörfer 99], [Wang 99].

In this paper, we propose a new and elegant technique of merging LBIST and external test patterns. The idea is to concurrently fill one subset of the scan chains with pseudo-random test data from the PRPG and the remaining scan chains with deterministic test data from an external tester. We do this in a round robin fashion such that each subset of scan chains receives deterministic data during the application of the complete hybrid test set. We refer to these test patterns as "hybrid test patterns" since they are part pseudo-random and part deterministic. The hybrid test patterns can be applied using the STUMPS architecture with a minor modification (as will be described in Sec. 2). The approach is entirely orthogonal to the earlier approaches to increase LBIST fault coverage (i.e., test point insertion or TPG modification). If needed, any of the techniques in [Muradali 90], [Venkataraman 93], [Hellebrand 95], [Tamarapalli 96], [Touba 96], and [Tsai 97] can be used in the proposed method to further boost fault coverage.

Hybrid patterns represent a technique of getting the most out of pseudo-random and deterministic test data by merging both into a single pattern. Pseudo-random LBIST patterns require minimal or no tester memory but have poor fault coverage. ATPG generated deterministic patterns, on the other hand, achieve the required fault coverage but require excessive tester memory. Hybrid patterns can detect a large number of random pattern resistant faults that are typically missed by LBIST. Random pattern resistant faults tend to be in spatial clusters that are fed by nearby scan elements. By grouping together structurally related scan chains and applying deterministic data to them, hybrid patterns provide much better fault coverage than LBIST patterns and yet require only a fraction of the tester memory storage required by full deterministic patterns. This leads to the following significant advantages:

1. *Hybrid patterns provide a large reduction in the top-up test pattern data volume.* There is a significant reduction in the number of top-up deterministic test

patterns required since hybrid patterns can detect far more faults than pure LBIST patterns. Reduction in top-up test patterns addresses the critical issue of over-loading of the tester memory.

2. *Reduction in the amount of pseudo-random data applied.* The number of hybrid patterns required for a circuit-under-test is orders of magnitude less than the number of LBIST patterns required. This addresses LBIST power issues.

3. *Hybrid patterns are compatible with current design flows.* Hybrid patterns can be applied with a minor modification to the STUMPS architecture. Also, hybrid patterns can be generated with currently available commercial ATPG software.

The paper is organized as follows: Sec. 2 reviews the STUMPS based LBIST architecture and explains our proposed hybrid test patterns. Sec. 3 gives experimental results showing the effectiveness of the proposed test patterns. Sec. 4 concludes the paper.

2. Proposed Hybrid Test Patterns

To explain the proposed hybrid test patterns, we have chosen the STUMPS LBIST architecture. We first review the STUMPS architecture in order to better explain the hybrid test patterns.

2.1 Review of STUMPS Architecture

The *self-testing using MISR and parallel SRSG* (STUMPS) architecture was first proposed in [Bardell 82]. MISR stands for multiple input shift register. SRSG is equivalent to PRPG, pseudo-random pattern generator. It was originally applied at the board level and subsequently at the chip level. The STUMPS architecture fits in well with all designs that have scan as the basic DFT feature. There is a centralized and separate LBIST hardware comprising of PRPG, MISR and LBIST controller. The most commonly used PRPG is the linear feedback shift register (LFSR). The scan paths are driven in parallel by the PRPG and the signature generated in parallel from each scan path using the MISR. An LBIST architecture based on STUMPS is shown in Fig. 1.

The LBIST controller is comprised mainly of a pattern counter and a shift counter. The pattern counter keeps track of the number of LBIST patterns applied. The total number of LBIST patterns applied depends upon the test time allotted to the LBIST operation. The shift counter keeps track of the number of cycles required to fill the scan chains with pseudo-random test data. The number of cycles required is equal to the sum of the length of the longest scan chain plus the number of cycles in a capture window. There will be a simple addition to the LBIST controller to generate the proposed hybrid test patterns.

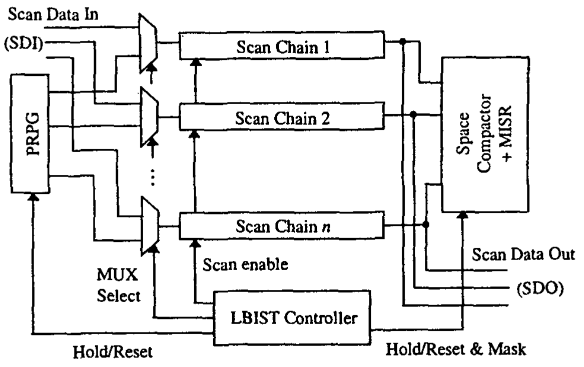

Figure 1. STUMPS Based LBIST Architecture

2.2 Modifications to the LBIST Controller

Assume there are n scan chains in a design. Each scan chain has a MUX at the scan input that determines whether the scan data comes from the PRPG or the external tester (see Fig. 1). In the normal LBIST approach, the select lines of the MUXes are driven by the LBIST controller to choose the data from the PRPG for the entire LBIST session. In our proposed hybrid test approach, the LBIST controller is modified to drive the select lines such that for each hybrid test pattern applied we have partial pseudo-random test data (for n-m scan chains) and partial deterministic test data (for the remaining m scan chains). While the PRPG is shifting pseudo-random data into (n-m) scan chains, the external tester shifts deterministic data into the other m scan chains.

In normal LBIST, the MUXes have a common select signal from the LBIST controller that chooses data from the PRPG during the LBIST session and data from the external ATE for the top-up patterns. For generating the proposed hybrid patterns, each set of m MUXes has a separate select signal. Assume there are k sets of m scan chains, $k = n/m$. The technique to partition the n scan chains into k sets of m scan chains will be discussed in the next section. Let the sets of m scan cells be labeled $\boldsymbol{D_1}\ \boldsymbol{D_2}$ ... $\boldsymbol{D_k}$. The corresponding select signals are labeled Sel_1 Sel_2 ... Sel_k.

The pattern counter will now keep track of L, the number of hybrid patterns applied per set of scan chains. Let a value of 0 on the select signal choose the data from the external ATE and a value of 1 choose data from the PRPG. Thus, the value 011...11 on the k select signals (Sel_1 ... Sel_k) will shift test data from the external ATE into the m scan chains in $\boldsymbol{D_1}$ and test data from the PRPG into the remaining scan chains i.e., the remaining (n-m) scan chains not included in $\boldsymbol{D_1}$. The LBIST controller will output this value of 011...11 for L test patterns. For the next L test patterns, the select signals produced will be 1011...11 and so on. Thus, deterministic data is applied to D_1 ... D_k in a round robin fashion with L hybrid patterns applied to each set. This is shown diagrammatically in Fig. 2.

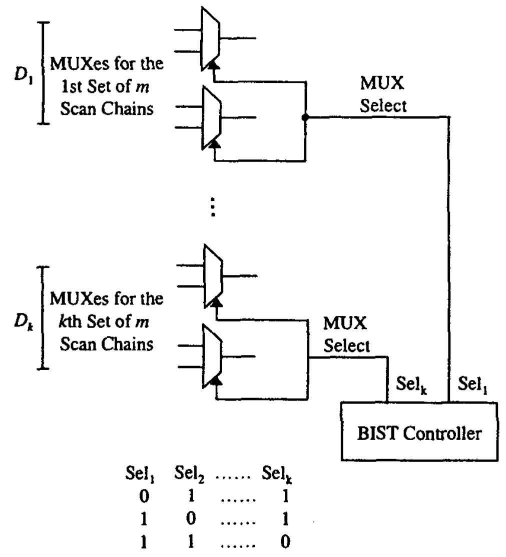
D_1
MUXes for the 1st Set of m Scan Chains
MUX Select
D_k
MUXes for the kth Set of m Scan Chains
MUX Select
Sel_k
Sel_1
BIST Controller
Sel_1 Sel_2 ...... Sel_k
0 1 1
1 0 1
1 1 0

Figure 2. LBIST Controller for Hybrid Test Patterns

2.3 Generation of Hybrid Test Patterns

We next explain two ways of generating the hybrid test patterns. One approach is to start with a full pseudo-random test pattern and do partial replacement of this pseudo-random data with deterministic data. The other approach is to start with a fully deterministic test pattern and do partial replacement with pseudo-random data.

<u>Approach 1</u>: Replacing part of pseudo-random pattern with deterministic data

In this approach, deterministic ATPG is performed for the m scan chains in the set D_i with prior knowledge of the pseudo-random data that will be applied to the remaining n-m scan chains. The LFSR is simulated to obtain the pseudo-random data that is shifted into the n-m scan chains. The steps in the proposed method are:

```
for each set $D_i$ of m scan chains {
  for L LFSR simulated test patterns {
    Take the portion of each LFSR simulated test
    pattern corresponding to the remaining (n-m) scan
    chains and input these as ATPG constrained values
    to the ATPG software. Perform deterministic
    ATPG on the design. (This will yield a
    deterministic test pattern for the scan chains
    included in $D_i$, which will target random pattern
    resistant faults that are detectable with the pseudo-
    random values in the other scan chains.)
  }
}
```

This will generate a total of $L \times k$ hybrid test patterns ($k = n/m$), since there are k sets of m scan chains and L hybrid test patterns are generated for each.

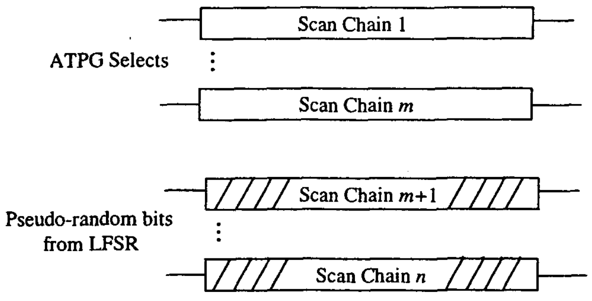

Figure 3. Generation of Hybrid Test Patterns

The hybrid test patterns generated by this approach will have high fault coverage since efficient deterministic ATPG can be done with the prior knowledge of the pseudo-random data that will be applied to the other scan chains. Knowledge of the pseudo-random data will considerably prune down the solution space that the ATPG targets.

<u>Approach 2</u>: Replacing part of deterministic pattern with pseudo-random data

The second approach is to start with a deterministic test pattern set, T, for the design. The hybrid test set will be generated by partial replacement of deterministic data with pseudo-random data in T. The large top-up test set needed after LBIST is due to the fact that LBIST patterns detect a very minor percentage of random-pattern resistant or hard-to detect faults. Hybrid test patterns will greatly reduce the top-up test set if they detect a large number of the hard faults. Thus, the initial deterministic test pattern set, T, should be generated by targeting only the hard faults in the circuit.

The steps in this approach are:

```
for each pattern in T {
  Generate pseudo-random data by the simulation of
  a LFSR.
  for each set $D_i$ of m scan chains {
    Replace deterministic data for the remaining (n-m)
    scan chains with the LFSR simulated pseudo-random
    data.
```

```
        Fault simulate the hybrid pattern obtained against the
        hard faults in the circuit to get the number of hard
        faults detected by the hybrid pattern.
    }
    Keep the hybrid pattern that detects the maximum
    number of hard faults.
}
```

The computation time required by this approach is mostly determined by the computation time required to generate the initial deterministic test set since fault simulation requires negligible computation time as compared to ATPG.

2.4 Parameters of Hybrid Test Patterns

There are two primary variables involved in the proposed hybrid test pattern scheme. One parameter is *m*, the number of scan chains having deterministic test data per hybrid test pattern. The other parameter is *L*, the number of hybrid patterns applied per set of scan cells. A discussion of these parameters follows.

We have concurrent pseudo-random data for (*n-m*) scan chains and deterministic patterns for the remaining *m* scan chains. The value of *m* chosen will depend primarily upon the number of channels from the tester, and the grouping of scan chains into blocks of *m* will depend on the design.

Testers have a limited number of channels designed to drive scan chains. These channels have greater depth than channels designed to drive functional pins. Some tester memories are configurable where the depth of the memory can be increased at the expense of width. The tester configuration will determine the optimal choice of *m* that will save maximum tester memory. Say the tester has 4 channels to drive scan chains. One option is to choose *m*=4, the tester channels would drive each set of 4 scan chains. *m* should not be chosen too large since that will increase the tester memory requirement of the hybrid patterns.

In dividing the *n* scan chains into sets of *m*, it is best to group together scan chains that are most structurally related. This aids the ATPG tool in detecting a random pattern resistant fault since it can deterministically control most of the scan chains that drive the cone of logic that contains that random pattern resistant faults. Consider an example circuit with 8 scan chains. Consider the case in Fig. 4 where scan inputs from three scan chains (scan chains 1, 3 and 5) feed the combinational logic block that drives a scan element (the third scan element of scan chain 1). All faults in that cone of logic will definitely be detected by hybrid patterns if the three scan chains are grouped together. This is due to the fact that there will exist hybrid patterns with deterministic test data for these three scan chains merged with pseudo-random data for the remaining scan chains. The three scan chains can be grouped together if *m* is chosen to be equal to or greater than three. For *m*=2, the number of faults detected in the cone can be maximized by grouping together scan chains 1 and 3 since they control the greatest number of the inputs to the combinational cone.

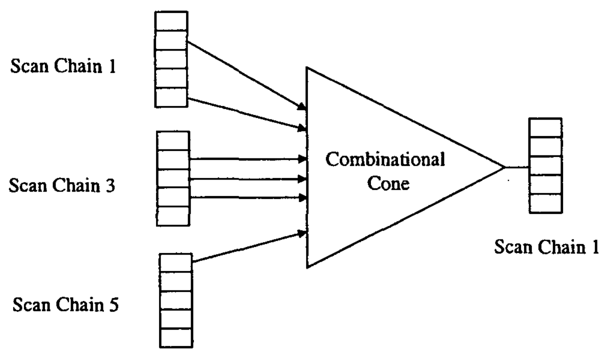

Figure 4. Scan Chain Grouping Based on Structural Dependencies.

Two approaches are possible for grouping the scan chains. If scan elements are stitched into scan chains according to their functional use, it is relatively easy to group scan that are structurally related. Scan chains belonging to the same functional block could be grouped together into one set. For example, all scan chains in the memory management block could be grouped into one set.

The other possible approach is to use the structural information of the circuit to group the scan chains. A scan chain dependency matrix can be formed by tracing each scan element backward to find the scan inputs that feed the cone of logic that drives that scan element. The scan chain dependency matrix gives information about the scan chains that are inputs to the cone of logic that drives each scan element. The rows in the matrix correspond to each scan element and the columns to the scan chains. An entry c_{ij} in the matrix denotes the number of scan elements of scan chain *j* that are inputs to the cone of logic that drives the scan element *i*.

Thus, two scan chains are closely related structurally if there exists a large number of rows in the scan dependency matrix where the entries in the columns corresponding to the two scan chains are both large. From the scan chain dependency information of the scan elements, a clustering can be done which aims at grouping together scan chains that are closely related structurally. Any standard clustering algorithm can be used. The cost function that has to be maximized is the amount of structural sharing between scan chains in the same cluster.

The other parameter of hybrid test patterns is *L*, the number of hybrid patterns applied per set of scan chains.

Hybrid patterns require much less tester memory for storage as compared to full deterministic patterns but they do require some tester memory. Thus, the optimal value of L would be the point where the increase in fault coverage of random pattern resistant faults obtained by additional hybrid patterns tapers off and can no longer be considered significant.

Instead of applying a fixed number of hybrid test patterns to each scan chain set, it would be more efficient to optimally choose the number of hybrid test patterns applied to each set. Deterministic ATPG for each set of scan chains can be done until there is no appreciable increase in fault coverage obtained by additional hybrid test patterns. The hybrid test pattern set thus generated would be more efficient as compared to the hybrid set generated with a fixed number of patterns per scan chain set. The cost will be in terms of hardware. Instead, of having to store just L, the LBIST controller would have to store the specific number of patterns applied for every set of scan chains.

With hybrid patterns, each test vector stored in the tester memory now has data for m scan chains rather than n scan chains. This is a significant reduction. As an example, the PowerPC core design has around 8,500 scan elements that are organized approximately into 50 scan chains [Pressly 99]. Assuming m=8, a hybrid test pattern will be approximately 15% of a full deterministic test pattern. For the ASICs in [Hetherington 99] that have 128 scan chains, a hybrid test pattern will be as low as 6.25% of a full deterministic test pattern. A full LBIST test pattern requires no tester memory, however we have seen that pseudo-random test patterns alone are limited in terms of the fault coverage they can achieve. The proposed hybrid test patterns represent a technique of getting the most out of pseudo-random and deterministic test data. Experimental results are presented in the next section.

3. Experimental Results

The results presented in [Pressly 99] are used as a starting point for our experiments. Pressly, *et al.*, have presented the findings of implementing LBIST on a PowerPC embedded core microprocessor. Some approximate statistics of the version one G2 PowerPC core used in the experiments are: 270K gates in the netlist, 1,168,490 uncollapsed faults, 8,500 scan elements currently stitched into 51 scan chains, no test point insertion [Pressly 99].

Pressly, *et al.*, have presented a feasibility study on the benefits and trade-offs involved in the integration and use of a LBIST mechanism for PowerPC embedded core microprocessors. We are interested in one aspect of their findings, namely, the reduction in tester storage requirement resulting from LBIST. The LBIST flow used for the PowerPC core is the common LBIST flow explained in the first section. LBIST is done for a fixed number of cycles and then deterministic ATPG is done to top-up the LBIST fault coverage. We reproduce the results presented in [Pressly 99] on reduction of tester memory storage in Table 1. Tester storage requirement is computed as the number of scan patterns times the number of scan elements. After 500K LBIST cycles, reduction in tester memory storage requirement is around 30%.

Table 1. Reduction in Tester Memory Storage Requirements from LBIST [Pressly 99]

Number of LBIST Patterns	Number of Top-Up Det. Patterns	Tester Storage Req. for Top-Up Patterns (Mbits)	Reduction in Tester Storage Req. (Mbits)	% Reduction of Storage Req.
0	8,200	69.70	0	0
5K	7,924	67.35	2.35	3.3
50K	7,011	59.59	10.11	14.5
500K	5,625	47.81	21.89	31.4

Details of the experiments to generate hybrid patterns are described below:

Grouping of scan chains: The structural information of the circuit was used to group the scan chains. Each scan element was traced back to find the scan inputs that feed the cone of logic that drives that scan element. From the scan chain dependency information for all of the scan elements, a grouping of scan chains is done that maximizes the number of scan elements whose fanin scan chains belongs to the same set.

Generation of initial deterministic pattern set: The approach used for generating hybrid patterns is the second approach described in Sec. 2.2. Recall that the hybrid test patterns will be most effective if the initial deterministic test set, T, is generated by targeting hard faults. In this experiment, the last 3500 patterns of the top-up test pattern set generated after 100K LBIST cycles is used as T. The 100K LBIST cycles detect most of the easy-to detect faults.

Generation of initial hybrid test pattern set: There are six sets in our circuit. For each set, we took the test set T and replaced deterministic data for all the scan chains that are not in the set with pseudo-random data (generated by the simulation of the LFSR). This gave us an initial hybrid test set that is six times the size of T (6*3500 = 21000 patterns).

Table 2. Comparison of Tester Storage Requirement

	Num. of Initial Patterns	Tester Storage Requirement of Initial Patterns (Mbits)	Num. of Top-Up Deterministic Patterns	Tester Storage Requirement of Top-Up Patterns (Mbits)	Reduction in Tester Storage Requirement (Mbits)	Percentage Reduction of Storage Requirement
Full ATPG	0	0	8,200	69.70	0	0
LBIST + ATPG	500,000 LBIST	0	5,625	47.81	21.89	31.4%
Hybrid + ATPG	**6,378 hybrid**	**9.04**	**3016**	**25.64**	**35.02**	**50.2%**

Generation of final hybrid test pattern set: We then fault simulated, with fault dropping, the hybrid patterns obtained against the hard faults in the circuit (i.e., the faults remaining after 100K LBIST cycles). We dropped all patterns that did not detect any faults. Total number of hybrid patterns remaining after this step was 6378. One pass of fault simulation with fault dropping can make the number of hybrid patterns retained for each set uneven. The set whose corresponding hybrid set was processed first would retain the maximum patterns and so on. A couple of passes with different orders of processing the sets was needed to bring the number of hybrid patterns retained per set to be almost equal.

Optional final step: Let the top-up test pattern set generated after the hybrid set be *P*. A second pass can further increase the efficiency of the hybrid test pattern set with the initial deterministic test pattern set now as *P*.

Reduction in external ATE test data:
Number of hybrid test patterns = 6378
Number of deterministic top-up test patterns = 3016
Tester storage requirement of the top-up test pattern set = 3016*8500 = 25.64 Mbits
Final size of external tester storage required = 9.04+25.64 = 34.68 Mbits
Reduction in Tester Storage Requirement = 69.70-34.68 = 35.02 Mbits

The experimental results clearly indicate greater reduction in tester storage requirement (50%) as well as a major reduction in the cycles of pseudo-random data applied (6378). We strongly feel that even better results can be obtained by further adapting the parameters of the hybrid patterns to the design.

4. Conclusion

This paper presents the concept of hybrid test patterns, where deterministic data and pseudo-random data are applied concurrently. Techniques for generating hybrid test patterns are described. Experimental results on a PowerPC core show that a significant reduction in test data volume can be achieved as compared to conventional LBIST followed up by top-up patterns. A detailed discussion of the theory behind hybrid patterns is presented. The parameters of hybrid patterns and approaches for their generation can be adapted to the design and design flow to generate maximum reduction in test data volume.

The work presented in the paper addresses the issue of tester memory overload, which is of prime significance in the current industry scenario. Hybrid test patterns provide a large reduction in the top-up test patterns data volume. Hybrid test patterns also reduce the cycles of pseudo-random data applied by orders of magnitude as compared to conventional LBIST, thus addressing the power issues arising with using large number of LBIST cycles.

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, Nov. 1982.

[Bardell 87] Bardell, P.H., W.H. McAnney, and J. Savir, "Built-In Test for VLSI: Pseudorandom Techniques," *John Wiley and Sons*, New York, 1987.

[Bottoms 98] Bottoms, B., "The Third Millennium's Test Dilemma," *IEEE Design & Test of Computers*, Vol. 15, No. 4, pp. 7-11, Fall 1998.

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

[Gerstendörfer 99] Gerstendörfer, S., and H-J Wunderlich, "Minimized Power Consumption for Scan-Based BIST," *Proc. of International Test Conference*, pp. 77-84, Sept. 1999.

[Hellebrand 95] Hellebrand, S., J. Rajski, S. Tarnick, S. Venkataraman, and B. Courtois, "Built-In Test for Circuits with Scan Based on Reseeding of Multiple-Polynomial Linear Feedback Shift Registers," *IEEE Transaction on Computers*, Vol. 44, No. 2, pp. 223-233, Feb. 1995.

[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.

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

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

[Tamarapalli 96] Tamarapalli, N, and J. Rajski, "Constructive Multi-Phase Test Point Insertion for Scan-Based BIST," *Proc. of International Test Conference*, pp. 649-658, 1996.

[Touba 96] Touba, N.A., and E.J. McCluskey, "Test Point Insertion Based on Path Tracing," *Proc. of VLSI Test Symposium*, pp. 2-8, 1996.

[Tsai 97] Tsai, K.-H, S. Hellebrand, J. Rajski, and M. Marek-Sadowska, "STARBIST: Scan Autocorrelated Random Pattern Generation," *Proc. Design Automation Conference*, pp. 472-477, June 1997.

[Venkataraman 93] Venkataraman, S., J. Rajski, S. Hellebrand, and S. Tarnick, "An Efficient BIST Scheme Based on Reseeding of Multiple Polynomial Linear feedback Shift Registers," *Proc. of International Conference on Computer-Aided Design*, Vol. 9, No. 6, pp. 584-593, Jun. 1990.

[Wang 99] Wang, S., and S.K. Gupta, "LT-RTPG: A New Test-Per-Scan BIST TPG for Low Heat Dissipation," *Proc. of International Test Conference*, pp. 85-94, Sept. 1999.