Low-Power Weighted Pseudo-Random BIST Using Special Scan Cells

Shalini Ghosh[1], Eric MacDonald[2], Sugato Basu[3], and Nur A. Touba[1]

[1]Dept. of Electrical & Computer Engg.
University of Texas at Austin
Austin, Texas 78712-1188
{shalini,touba}@ece.utexas.edu

[2]Dept. of Electrical & Computer Engg.
University of Texas at El Paso
El Paso, Texas 79968-0523
emac@utep.edu

[3]Dept. of Computer Sciences
University of Texas at Austin
Austin, Texas 78712-1188
sugato@cs.utexas.edu

ABSTRACT

In this paper, a technique for weighted pseudo-random built-in self-test (BIST) of VLSI circuits is proposed, which uses special scan cells and a new weight selection algorithm to achieve low power dissipation. It is based on weighted pseudo-random scan testing in which only 3 weight values are used – 2 fixed values (0 or 1) and 1 random value (0.5). A new weight selection algorithm is used to select a set of weights that achieves high fault coverage while reducing power. The idea is to minimize power by careful selection of the set of scan cells having fixed values (0 or 1) in order to reduce switching activity. To implement this in hardware, a new scan cell design is proposed that can do scan and capture in the normal mode as well as fixed-bit mode. The new scan cell hardware increases the area of a typical circuit by less than 4%, but reduces power by as much as 96%, as indicated in experiments performed on benchmark circuits.

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

General Terms: Reliability

Keywords: Low power, built-in self-test, weighted pseudo-random testing

1. INTRODUCTION

Built-in self-test (BIST) involves performing test pattern generation and output response analysis using on-chip circuitry. The most economical BIST techniques are based on pseudo-random pattern testing. There are two well-known problems with pseudo-random BIST: low fault coverage and high power dissipation. Low fault coverage arises due to the presence of random pattern resistant (r.p.r.) faults [4], which have low detection probabilities. Solutions to this problem involve either modifying the circuit-under-test (CUT) by inserting test points to increase the detection probabilities, or by modifying the test pattern generator so that it generates patterns that detect the r.p.r. faults. The problem of high power dissipation comes from the fact that pseudo-random patterns cause much greater switching activity in the CUT than what occurs during normal functional operation. This can result in overheating, as the chip package may only be capable of handling the power dissipation that occurs during functional operation. Moreover, for portable electronics where BIST is used out in the field, it is desirable that the BIST use a minimal amount of energy to preserve battery life.

The problem of low fault coverage for pseudo-random BIST has been studied for a long time and quite a number of solutions have been proposed. One of the most attractive involves adding weight logic to bias the pseudo-random patterns towards those that detect the r.p.r. faults. A number of weight selection algorithms have been proposed for finding a minimal number of weight sets to achieve a desired fault coverage [3] [12].

The problem of reducing power dissipation during BIST is a more recent problem that has gained attention. Some interesting techniques have been proposed including the following: dual-speed linear feedback shift register (DS-LFSR) [14], low transition random test pattern generation (LT-RTPG) [15], scan output holding [6], test vector inhibiting [7], circuit partitioning [8], and modified clocking schemes [9].

An attractive approach that combines weighted pattern testing with a low power BIST was proposed by Wang in [16]. This approach can achieve high fault coverage by targeting random pattern resistant fault while still reducing power dissipation compared with conventional weight pattern testing. Three things are combined in Wang's method: LT-RTPG, 3-valued weights, and scan re-ordering.

This paper presents a new approach for combining weighted pattern testing with low power BIST. A different hardware

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
GLSVLSI '04, April 26–28, 2004, Boston, Massachussets, USA
Copyright 2004 ACM 1-58113-853-9/04/0004…$5.00.

scheme and weight selection methodology is used. Compared with Wang's approach [16], the proposed technique provides better scalability and power reduction.

The rest of the paper is organized as follows: Section 2 gives background on the 3-valued weight set generation algorithm of Pomeranz *et al.* [12], which we have improved upon; Section 3 describes our proposed low power 3-valued weight set generation algorithm; Section 4 describes a special scan cell design that enables us to implement our low power weighted pseudo-random testing algorithm in hardware, along with an area-overhead analysis and simulation study; Section 5 gives the results of experiments on ISCAS-89 benchmark circuits, followed by the conclusion and possible future work in Section 6.

2. BACKGROUND

Pomeranz *et al.* [12] introduced an algorithm for weighted pseudo-random testing, where the weight sets generated were *3-valued*, i.e., each weight value is 0, 0.5 or 1. We have modified this algorithm to generate weight sets that reduce the power dissipation during testing. Note that by *weight set*, we will refer to a vector in which each position can be 3-valued – such a weight set will be used to generate weighted pseudo-random test vectors.

The basic idea of the 3-valued weight set generation algorithm of Pomeranz *et al.* is to use a deterministic test set to find a small number of weight sets that can be used to generate weighted random test patterns for a circuit. The goal is to get significant fault coverage using a much smaller number of weights than the number of deterministic test vectors, thereby making the scheme useful for built-in self-test (BIST) due to its small area requirement. The test generation process starts with a set of purely random patterns to detect the easy-to-detect faults. After that, weight sets are computed by combining test vectors from the deterministic test vector set. Two test vectors *v1* and *v2* are combined using the following method: if *v1* has a value 1 (0) in a bit position, then the corresponding bit in the combined vector gets the value 1 (0) if *v2* has the same bit value 1 (0) or the value X at that position. If at any position *v1* has a bit value 1 (0) and *v2* has a different bit value 0 (1) at that position, the corresponding position in the combined vector gets a value R. Note that X signifies an unspecified bit value, while R indicates a weight value of 0.5. For example, combining **XX**01100 and **X**00**X**001 would give **X**001**R**0**R**.

Once a weight set is created, the algorithm generates *N* weighted random patterns using the LFSR by fixing the scan cell positions corresponding to 0 or 1 weight values in the weight set and randomly changing scan cell positions having a weight value of 0.5. If any bit position in the weight set has a value of X, it is filled by minimum-transition (MT) fill [13]. In MT-fill, a series of X entries in a vector are filled with the same value as the first non-X (and non-R) entry on the right side of this series. This minimizes the number of transitions when such a vector is scanned into a scan chain. For example, consider that after the combination process described earlier, we get a weight set 1**X**0**R**0**R**10**X**1**X**0. This set, after MT-fill, would become 100**R**0**R**101100. If the set has a sequence of X bits that is not terminated by a non-X bit on the right side, then it is filled by the bit value to the left of the sequence. For example: 10**R**0**RR**1011**XX** will become 10**R**0**RR**1011**11** after MT-fill.

For each weighted random pattern, the algorithm performs fault simulation and removes the detected faults from the fault-list. When no faults are detected for one run of *N* weighted random patterns for a weight set, it decreases the number of random values in the weight set *(K)* by 1, i.e., it fixes one more bit in the next weight set, in order to detect the harder-to-detect faults.

Our main motivation of using a 3-valued weight set generation algorithm is the fact that weight values of 0 or 1 can be fixed throughout generation of the weighted random patterns for a weight set and thereby reduce switching activity, and hence power. So, if we have hardware support to fix bit values in the scan cells, we can use a modified version of the 3-valued weight set generation algorithm tuned towards generating weight sets that fix bits corresponding to scan cell positions giving maximum power savings. Accordingly, the combinational part of the circuit connected to the scan cells having the fixed bit values would not have any transitions during application of the weighted random patterns, resulting in reduced power consumption during testing. We modified the 3-valued weight set generation algorithm to make it power-sensitive, and modified the SFN cell design of AlShaibi *et al.*[1] to create fixed-mode scan cells that enable us to implement our scheme in hardware.

3. ALGORITHM DESCRIPTION

3.1 Low-power weighted pseudorandom testing

Our algorithm takes the basic idea of the original 3-valued weight selection algorithm of Pomeranz *et al.*, but makes an important change in the weight set generation step that reduces power consumption during testing.

In the original algorithm, a weight set is selected to maximize coverage. First, the faults are ordered in decreasing order of how hard-to-detect they are, estimated by the number of deterministic test vectors that cover the fault (the less the number of vectors covering a fault, the more it is hard-to-detect). Then, two test vectors are selected, each of which covers one hard-to-detect fault and also has high fault coverage. A weight set is created by taking the combination of these two vectors (details are shown in the *FindIntersection( )* function in Figure 1 and explained in Section 2). The weight set is combined with other high fault-coverage test vectors covering hard-to-detect faults, and the number of fixed bit positions in the weight set decreases at each step (implying that the number of random bit positions increase). This is continued till the number of random bit positions in the weight set does not exceed the desired value *K*.

INPUTS: ***F*** = target faults to be covered, ***C*** = minimum accepted fault coverage, ***K*** = initial number of random values in weight set, ***R*** = number of pure random patterns generated before weighted pattern generation, ***P*** = vector of estimated power saved by fixing each scan cell position at 0 or 1, ***N*** = number of weighted pseudo-random test patterns generated for each weight set, ***D*** = deterministic test set
OUTPUT: ***W*** = set of 3-value weight sets

WeightSetGeneration( ):

1. Let LFSR generate *R* pure random patterns, remove detected faults from *F*. Initialize set *W* as empty
2. Let originalNumberOfFaults = number of faults in *F*
3. While (number of current faults in *F*) > (1-***C***) * originalNumberOfFaults, repeat Step 4-6
4. Using *LowPowerWeightSelection( ),* generate weight set *w* having *K* random bits. Add *w* to set *W*
5. Generate *N* weighted random patterns using LFSR, by fixing scan cells positions with 0 or 1 weight value in *w*, and randomly changing scan cells having 0.5 values in *w*. For each random pattern, perform fault simulation and remove detected faults from *F*
6. If no fault is detected in Step 4, set *K* = *K* - 1
7. Return *W*

LowPowerWeightSelection(K):

1. For every fault in the set *F* of undetected faults, find the test vectors from *D* that cover the fault
2. Select a fault *f* that has minimum number of test vectors covering it. Among all test vectors that detect *f*, find the vector *t* that covers most number of faults
3. Mark vector *t*, and initialize weight set *w* = *t*
4. While (number of random bit positions in *w*) < *K*, repeat steps 5-6
5. Select the unmarked test vector *t'* from *D* that has the maximum *PowerSavedScore( )* with *w*
6. Set *w* = *FindIntersection(w,t').* Mark *t'*
7. Return *w*

PowerSavedScore(w,t'):
Initialize *saved*=0. For all positions *p* where *w* and *t' have* same bit value *b*, set *saved* = *saved* + (power saved by fixing position *p* at *b*, obtained from *P*). Return *saved*

FindIntersection(w,t'):

Initialize *intersection* to have R (i.e., 0.5) in all bit positions. For every position where both *w* and *t'* have same bit value *b* (0 or 1) or one has *b* and the other has X, set that position in *intersection* to *b*. Return *intersection*

Fig 1. Low-power weight-set selection algorithm

In our proposed low-power weight generation algorithm outlined in Figure 1, we make this weight selection scheme power-sensitive. First, for each scan cell position in the circuit, we estimate the power saved if that position were to be fixed at a value of 0 or 1, details of which are outlined in Section 3.2. Then we select the highest-coverage vector for the hardest-to-detect fault, similar to the scheme of Pomeranz *et al.* After that, instead of creating a weight set by choosing test vectors with high coverage, we select test vectors that will give maximum power saved (due to bit positions being fixed) when combined with the existing weight set. This ensures that during weight set creation, we fix those bits in the scan cell that will give us maximum power saved when we do weighted random pattern testing.

3.2 Estimating power saved by bit fixing

Wang *et al.* [14] has an elaborate method of estimating how much power is saved by fixing a particular circuit node to 0 or 1, by estimating the transition density for that node. However, calculating their metric for large circuits is time-intensive. We use a simulation-based approach that is fast and reasonably accurate for our purpose. We generate 100 random patterns and apply that to the circuit, calculating the average power dissipated per random pattern (*pR*) using a power simulation software. We then fix a scan cell position *i* to 1 (or 0) and calculate the average power dissipated by applying the same set of 100 random patterns, keeping scan cell *i* fixed at 1 (or 0) – the difference of this value and *pR* gives us an estimate of the power saved in the circuit by fixing the bit position *i* to 1 (or 0), which we denote *p1* (or *p0*) for position *i.* The values of *p1* and *p0* are estimated for each scan cell position in the circuit, giving us an estimate of the power saved in the circuit by fixing the scan cell positions to 1 and 0 respectively. These estimates are stored in vector *P* in Figure 1.

4. HARDWARE DETAILS

In this section, we will present the design of a fixed-bit scan cell that is motivated by the SFN cell design of AlShaibi *et al.* [1]. We will demonstrate its correctness using simulation results and also analyze the area overhead of this new scan cell design. Note that Pomeranz *et al.* [12] implemented bit fixing in their scheme by using a bit-fixing control logic between the output of normal scan cells and the combinational parts of the circuit. The problem with this approach is that the bit-fixing logic becomes quite complex for large circuits, incurring area overhead as well as delay overhead in the normal operation of the circuit. This prompted us to generalize the design of the scan cells to support bit fixing, so that our proposed approach scales well to large circuits.

4.1 Fixed-bit scan cell design

The SFN (Scan-Fixed-Normal) cell design of AlShaibi *et al.* is a scan cell capable of scanning-in values in the normal mode as well as the fixed mode. The normal mode operation is same as a typical scan cell. In the fixed mode, the output of the SFN cell that is

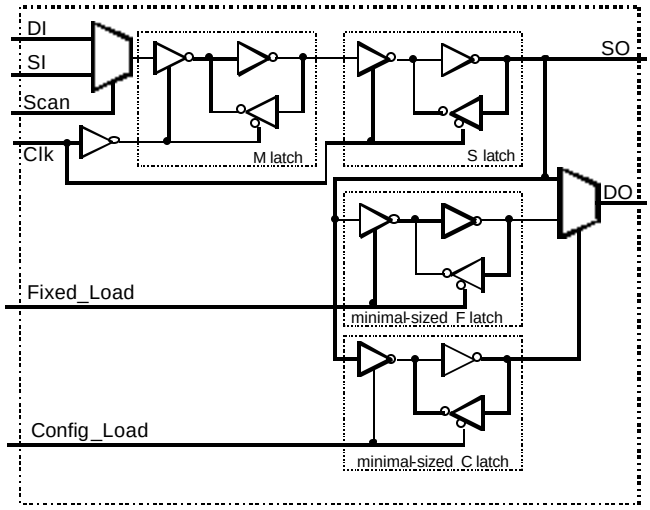

Fig 2. SFNC Scan Cell Design

connected to the combinational part of the circuit is held constant while values are scanned through the SFN cell in the scan chain, thereby reducing power consumption in the combinational circuit components connected to the SFN cell output.

A major improvement of our fixed-bit scan cell design is that apart from normal and fixed scan, it is also capable of performing the *capture* operation in both normal and fixed modes, something that AlShaibi *et al.* did not consider in their design. Their scheme was for test-for-clock architecture, whereas we consider the standard test-per-scan STUMPS architecture for BIST [2]. Scan capture is an essential operation of a scan cell in the standard STUMPS scan-BIST architecture, since here the scan chain captures the response of the circuit after application of the test vector. We call our design the SFNC (Scan-Fixed-Normal-Capture) cell, details of which are shown in Figure 2.

The SFNC has 6 inputs: apart from the usual *Data_Input (DI), Scan_Input (SI), Scan* signal and *Clock* signal, it had 2 other control signals – *Fixed_Mode* and *Config_Load*. The SFNC cell has 2 outputs: *Scan_Output (SO)* and *Data_Output (DO). DO* is connected to the combinational part of the circuit, while *SO* is connected to the *SI* of the next scan cell in the chain. Note that a normal scan cell has a single output line, which is connected to both the combinational part of the circuit (like *DO*) as well as to the scan-in of the next scan cell in the chain (like *SO*).

In our SFNC cell, the *M* and *S* latches together make a traditional flip-flop. The *C* latch holds the information whether the cell is fixed ($C = 1$) or not ($C = 0$), and the *F* latch holds the fixed bit value. When the SFNC cell is configured to be operating in the fixed mode so that $C = 1$, the output mux connects the output of *F* to *DO*. So, during scan and capture operations in this fixed mode, values shifted into the scan cell are shifted out through *SO*, but the combinational circuit connected to the cell sees the fixed bit output of the *F* latch through *DO*. So, in the fixed mode, there is no power dissipation in the combinational circuit connected to the output of the SFNC cell, since the *DO* output is held constant. In the normal mode, when $C = 0$, the SFNC cell behaves like a normal scan cell as *DO* gets driven by *SO*.

The values in the *F* and *C* latches must be scanned in separately in two different *n*-cycle scan sequences. In the first *n*-cycle sequence, the actual weight set is scanned in, which ends with asserting *Fixed_Load* for one cycle. This captures the output of the *S* latch into the *F* latch. The next sequence loaded is the configuration vector, which configures the *C* latches in each SFNC cell: it ends with a *Config_Load* assertion for one cycle loading the value held in the *S* latch into *C*. The configuration vector that is scanned in has a 1 corresponding to weight set positions having a fixed bit (0/1), and 0 corresponding to a weight set position having a random bit (0.5). For example, for the weight set 1XX110XX01, the configuration pattern would be 1001110011. Once the fixed values are loaded and the SFNC cells are configured, the LFSR is run to generate *N* pseudo-random test patterns. Each of these *N* random test patterns is weighted according to the weight set loaded into the scan chain.

At power-up of the circuit, the *C* latch in each SFNC cell needs to reset by the functional reset of the latches, preferably using asynchronous reset signals. This would make sure that the SFNC cell works in the normal mode at power-up. The *F* and *C* latches are minimum sized latches, i.e., their transistors are as small as possible, since these 2 latches drive small loads and are not timing-critical. The mux at the output would need to be large to drive the presented load. All latches in the design are level sensitive.

4.2 Estimation of area overhead

A standard scan cell has 2 typical latches, 1 mux, and 1 inverter, with 4 inputs and 1 output. Our SFNC scan cell design has 2 typical latches, 2 minimal-size latches, 2 muxes, and 1 inverter, with 6 inputs and 2 outputs. Let *X* be the typical latch area, *Y* be the inverter area, *Z* be the mux area and *X'* be the minimum latch area. So the percentage area increase of a SFNC cell over a standard cell would be $(2X' + Z) * 100 / (2*X + Y + Z)$.

Considering 130nm technology, typical values of *X, Y, Z* and *X'* would be 30, 7, 15 and 9 units respectively. So, the SFNC scan cell would be about 40% more in size compared to the standard scan cell. For example, consider a chip where 40% of the area is occupied by memory and 60% by logic, of which about 15% is for flip-flops. If all the flip-flops were replaced by SFNC flip-flops in the chip, the total chip area would increase by only 3.6%.

Our proposed STUMPS architecture with SFNC cells is shown in Figure 3. The overhead consists of an increase in the size of the scan cells (as explained above) and a ROM for storing the configuration bits and weight sets. Note that other BIST schemes that detect random pattern resistant faults also generally require a similar ROM, e.g., for storing weight sets for a weighted pseudo-random scheme [3], or for storing seed patterns in a LFSR re-seeding scheme [5], etc.

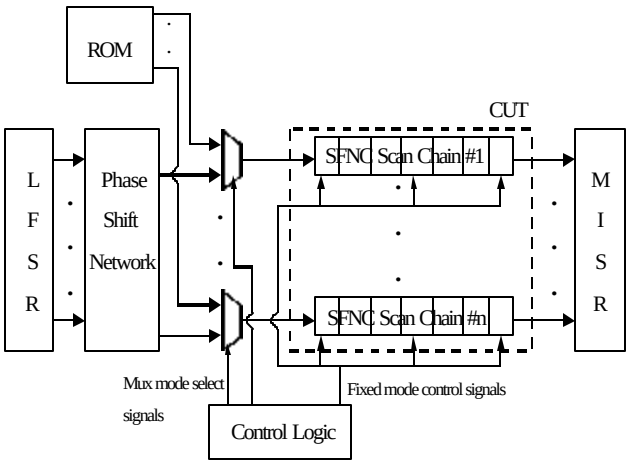

Figure 3. STUMPS architecture with SFNC cells

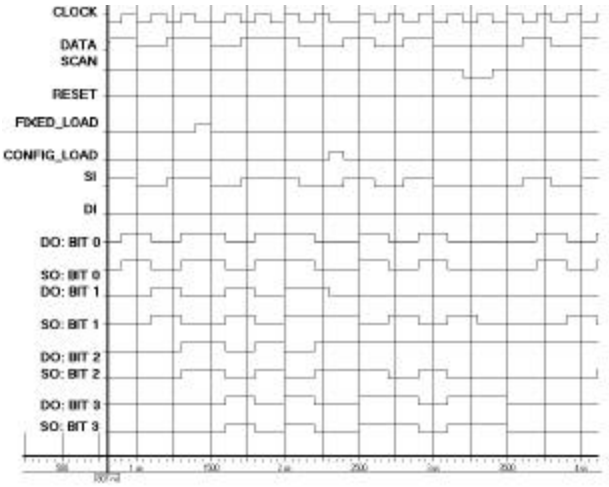

Figure 4. Waveform of test circuit

Apart from generating the usual scan testing control signals, the control logic shown in Figure 3 generates the extra control signals (e.g., *Config_Load*, *Fixed_Load*) for scanning in the configuration and weight sets. The mux at the input of the scan chains is needed to select either the next weight (or configuration) set from the ROM, or the next random pattern from the LFSR.

4.3 Simulation results

The correctness of the operation of the SFNC cell was verified by Verilog simulation on a test circuit that has four SFNC cells (numbered 0-3) functionally connected as a shift register. The waveforms are displayed in Figure 4. The weight set is scanned in the first four cycles: *SFNC-1* is fixed to 0 and *SFNC-2* is fixed to 1. Subsequently, *Fixed_Mode* is asserted. In the next four cycles, the configuration vector is scanned in: it configures *SFNC-0* and *SFNC-3* as normal scan cells, and *SFNC-1* and *SFNC-2* as fixed scan cells, followed by the assertion of *Config_Load*. From this point on, note that the data out of *SFNC-1* and *SFNC-2* in Figure 4 hold at their respective values, showing that the SFNC cell is performing correctly. We then do a scan in, capture and scan out – all the while the *DO* output of cells *SFNC-1* and *SFNC-2* remain unchanged and the *DO* outputs of cells *SFNC-0* and *SFNC-3* change with their corresponding *SO* values, as desired.

5. EXPERIMENTAL RESULTS

Experiments were performed on 4 standard ISCAS-89 benchmark circuits: s5378, s9234, s13207 and s15850. For each circuit, we used the ATALANTA toolkit [11] to generate the deterministic test vector set. We initially performed random test pattern generation to detect the easy-to-detect faults. In these experiments, an initial set of 1024 random patterns was generated. For designing weight sets to detect the remaining faults, we ran the *WeightSetGeneration* algorithm in Figure 1, with parameters N = 256 (i.e., corresponding to each weight set we generated 256 weighted random test patterns), F = number of detectable faults in the circuit, C = 0.98 (i.e., minimum accepted fault coverage on all detectable faults is 98%) and K = 0.10 * number of scan cells in the circuit (i.e., a maximum of 10% of the weight values were random, implying at least 90% of the weight values were fixed in each weight set).

The power dissipation for each test set was estimated by counting the number of node transitions in the whole circuit and weighting each node transition by the number of fanouts at the node. As can be seen from the results in Table 1, we get greater than 98% fault coverage for detectable faults on all circuits. For each weight set, we have to store the actual weight pattern and the configuration pattern in a ROM, but the number of weight sets is an order of magnitude less than the number of deterministic test vectors. So, our scheme is very well suited for a BIST environment. Table 1 also shows the ROM size required to hold the weight sets for each benchmark circuit. Each weight set requires 2 bits per scan cell (for the weight set and configuration sequence). The numbers in Table 1 assume no compression. Note, however, that the data stored in the ROM is highly compressible, so it is possible to reduce the size of the ROM considerably using an approach along the lines of what is used in [10]. The power reduction compared with the original weight set selection algorithm of Pomeranz *et al.* ranges between 84% to 96% on these benchmark circuits, showing that our scheme will be very effective in low-power BIST for power-critical applications.

Note that with comparable fault coverage, the power reductions we have obtained are more than Wang's approach [16], presumably because we have fixed bit positions more aggressively in our algorithm (at least 90% of the bit positions are fixed for each circuit). Moreover, our hardware design is more scalable for large circuits, since we don't have to design special-purpose decoding logic for weight generation for every new circuit, as is required for Wang's WR-BIST hardware [16].

Table 1. Results on low-power weighted pseudo-random testing algorithm on ISCAS-89 benchmarks

Circuit Name	#Scan Cells	#Test Vectors	Size of Weight Set	#Transitions with Algorithm in [12] & Normal Scan Cell	#Transitions with new Algorithm & SFNC Scan Cell	Fault Coverage	Size of Weight ROM (bits)	Power Reduction
s5378	214	118	12	7.11×10^8	2.49×10^7	99.58%	5K	96.49%
s9234	247	154	50	2.16×10^9	3.25×10^8	99.11%	25K	84.95%
s13207	611	240	29	4.65×10^{10}	6.85×10^8	98.45%	35K	95.85%
s15850	700	118	12	7.05×10^9	4.66×10^8	98.89%	17K	93.39%

6. CONCLUSIONS

In this paper we have proposed a new approach for combining weighted pseudo-random pattern testing with a low power BIST. Compared to Wang's solution [16], our algorithm achieves comparable fault coverage and gives more power reduction, while our hardware design is more generic and scalable.

In the future, we want to investigate other strategies of generating weight sets, based on not just power reduced but a combined score of faults coverage and power reduction, with a trade-off parameter between the two. This may enable us to achieve the desired fault coverage with smaller number of weight sets. We also want to investigate compression of weight sets [10] to reduce the size of the weight set ROM.

7. ACKNOWLEDGEMENTS

This material is based on work supported by the National Science Foundation under Grant No. CCR-0306238.

8. REFERENCES

[1] AlShaibi, M.F., and C. R. Kime. Fixed-biased Pseudo-random Built in Self-test for Random Pattern Resistant Circuits. *Proc. of International Test Conf.*, pp. 929-938, 1994.

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

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

[4] Eichelberger, E.B., and E. Lindbloom. Random-Pattern Coverage Enhancement and Diagnosis for LSSD Logic Self-Test. In *IBM Journal of Research and Development* 27(3), pp. 265-272, 1983.

[5] Krishna, C.V., A. Jas, and N. Touba. Test Vector Encoding Using Partial LFSR Reseeding. *Proc. of International Test Conf.*, pp. 885-893, 2001.

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

[7] Girard, P., L. Guiller, C. Landrault, and S. Pravossoudovitch. A Test Vector Inhibiting Technique for Low Energy BIST Design. *Proc. of VLSI Test Symposium*, pp. 407-412, 1999.

[8] Girard, P., L. Guiller, C. Landrault, and S. Pravossoudovitch. Circuit Partitioning for Low Power BIST Design with Minimized Peak Power Consumption. *Proc. of Asian Test Symposium*, pp. 89-94, 1999.

[9] Girard, P., L. Guiller, C. Landrault, S. Pravossoudovitch, and H.J. Wunderlich. A Modified Clock Scheme for a Low Power BIST Test Pattern Generator. *Proc. of VLSI Test Symposium*, pp. 306 –311, 2001.

[10] Jas, A., C.V. Krishna, and N.A. Touba, "Hybrid BIST Based on Weighted Pseudo-Random Testing: A New Test Resource Partitioning Scheme", *Proc. of IEEE VLSI Test Symposium*, pp. 2-8, 2001.

[11] Lee, H.K., and D. S. Ha. On the Generation of Test Patterns for Combinational Circuits. *Technical Report No. 12-93*, Department of Electrical Engineering, Virginia Polytechnic Institute and State University, 1991.

[12] Pomeranz, I., and S. M. Reddy. 3-Weight Pseudo-Random Test Generation Based on a Deterministic Test Set. *IEEE Trans. on Computer-Aided Design*, pp. 1050-1058, July 1993.

[13] Sankaralingam, R., R. Oruganti and N. Touba, Static Compaction Techniques to Control Scan Vector Power Dissipation. *Proc. of VLSI Test Symposium*, pp. 35- 42, 2000.

[14] Wang, S., and S. K. Gupta. DS-LFSR: A New BIST TPG for Low Heat Dissipation. *Proc. of International Test Conf.*, pp. 848-857, 1997.

[15] Wang, S., and S. K. Gupta. LT-RTPG: A New Test-Per-Scan BIST TPG for Low Heat Dissipation. *Proc. of lnternational Test Conf.*, pp. 85-94, 1999.

[16] Wang, S., Generation of Low Power Dissipation and High Fault Coverage Patterns for Scan-based BIST. *Proc. of International Test Conference*, pp: 834-843, 2002