Dinesh Babu al./ IJAIR *Vol. 2 Issue 7* *ISSN: 2278-7844*

High speed 3-Weight Pattern Generation based on ABIST

Nallapu Dinesh Babu[1], P.S.S.Chakravarthy[2]

[1]*Department of electronics and communication engineering, Narasaraopet Engineering college, Narasaraopet, Ap, India.*

[2]*proffesor Department of electronics and communication engineering, Narasaraopet Engineering college, Narasaraopet, Ap, India.*

dineshbabu434@gmail.com[1]

***Abstract*— We describe a method for on-chip generation of weighted test sequences for synchronous sequential circuits. For combinational circuits, three weights, 0, 0.5 and 1, are sufficient to achieve complete coverage of stuck-at faults, since these weights are sufficient to reproduce any specific test pattern. For sequential circuits, the weights we use are defined based on subsequences of a deterministic test sequence. Such weights allow us to reproduce parts of the test sequence, and help ensure that complete fault coverage would be obtained by the weighted test sequences generated.**

In this paper an accumulator-based 3-weight test pattern generation scheme is presented. The proposed scheme copes with the inherent drawbacks of the scheme proposed more precisely. First, it does not impose any requirements about the design of the adder i.e., it can be implemented using any adder design and then it does not require any modification of the adder; and hence it does not affect the operating speed of the adder. Furthermore, the proposed scheme compares favorably to the scheme proposed in terms of the required hardware overhead.

The weighted random test pattern generation represents a significant departure from classical methods of generating test sequences for complex large scale integration packages. The virtue of this technique is its simplicity and the fact that test-generation time is virtually independent of or gates in the logic package to be tested. This technique can be used both in a conventional tester and in a tester where the weighted random test pattern generation is implemented in hardware.

KEYWORDS: Built-in self test (BIST), test per clock, VLSI testing, weighted test pattern generation.

I. INTRODUCTION

Pseudorandom built-in self test (BIST) generators have been widely utilized to test integrated circuits and systems. The arsenal of pseudorandom generators includes, among others, linear feedback shift registers (LFSRs) [1], cellular automata [2], and accumulators driven by a constant value [3]. For circuits with hard-to-detect faults, a large number of random patterns have to be generated before high fault coverage is achieved. Therefore, weighted pseudorandom techniques have been proposed where inputs are biased by changing the probability of a “0” or a “1” on a given input from 0.5 (for pure pseudorandom tests) to some other value [10], [15]. Weighted random pattern generation methods relying on a single weight assignment usually fail to achieve complete fault coverage using a reasonable number of test patterns since, although the weights are computed to be suitable for most faults, some faults may require long test sequences to be detected with these weight assignments if they do not match their activation and propagation requirements. Multiple weight assignments have been suggested for the case that different faults require different biases of the input combinations applied to the circuit, to ensure that a relatively small number of patterns can detect all faults [4]. Approaches to derive weight assignments for given deterministic tests are attractive since they have the potential to allow complete coverage with a significantly smaller number of test patterns [10]. In order to minimize the hardware implementation cost, other schemes based on multiple weight assignments utilized weights 0, 1, and 0.5. This approach boils down to keeping some outputs of the generator steady (to either 0 or 1) and letting the remaining outputs change values (pseudo-) randomly (weight 0.5). This approach, apart from reducing the hardware overhead has beneficial effect on the consumed power, since some of the circuit under test (CUT) inputs (those having weight 0 or 1) remain steady during the specific test session [30]. Pomeranz and Reddy [5] proposed a 3-weight pattern generation scheme relying on weights 0, 1, and 0.5. The choice of weights 0, 1, and 0.5 was done in order to minimize the hardware implementation cost. Wang [8], [13] proposed a 3-weight random pattern generator based on scan chains utilizing weights 0, 1, and 0.5, in a way similar to [5]. Recently, Zhang *et al.* [9] renovated the interest in the 3-weight pattern generation schemes, proposing an efficient compaction scheme for the 3-weight patterns 0, 1, and 0.5. From the above we can conclude that 3-weight pattern generation based on weights 0, 1, and 0.5 has practical interest since it combines low implementation cost with low test time. Current VLSI circuits, e.g., data path architectures, or digital signal processing chips commonly contain arithmetic modules [accumulators or arithmetic logic units (ALUs)]. This has fired the idea of arithmetic BIST (ABIST) [6]. The basic idea of ABIST is to utilize accumulators for built-in testing (compression of the CUT responses, or generation of test patterns) and has been shown to result in low hardware overhead and lowimpact on the circuit normal operating speed [22]–[27]. In [22], Manich *et al.* presented an accumulator-based test pattern generation scheme that compares favorably to previously proposed schemes. In [7], it was proved that the test vectors generated by an accumulator whose inputs are driven by a constant pattern can have acceptable pseudorandom characteristics, if the input pattern is properly selected. However, modules containing hard-to-detect faults still require extra test hardware either by inserting test points into the mission logic or by storing additional deterministic test patterns [24], [25]. In order to overcome this problem, an accumulator-based weighted pattern generation scheme was proposed in [11]. The scheme generates test patterns having one of three weights, namely 0, 1, and 0.5 therefore it can be utilized to drastically reduce the test application time in accumulator-based test pattern generation. However, the scheme proposed in [11] possesses three major drawbacks: 1) it can be utilized only in the case that the adder of the accumulator is a ripple carry adder; 2) it requires redesigning the accumulator; this modification, apart from being costly, requires redesign of the core of the datapath, a practice that is generally discouraged in current BIST schemes; and 3) it increases delay, since it affects the normal operating speed of the adder. In this paper, a novel scheme for accumulator-based 3-weight generation is presented. The proposed scheme copes with the inherent drawbacks of the scheme proposed in [11]. More precisely: 1) it does not impose any requirements about the design of the adder (i.e., it can be implemented using any adder design); 2) it does not require any modification of the adder; and hence, 3) does not affect the operating speed of the adder. Furthermore, the proposed scheme compares favorably to the scheme proposed in [11] and [22] in terms of the required hardware overhead. This paper is organized as follows. In Section II, the idea underlying the accumulator-based 3-weight generation is presented. In Section III, the design methodology to generate the 3-weight patterns utilizing an accumulator is presented. In Section IV, the proposed scheme is compared to the previously proposed ones. Finally, Section V, concludes this paper.

TABLE I:TEST SET FOR THE C17 BENCHMARK

Test vector	Inputs A[4:0]
T1	00101
T2	01010
T3	10010
T4	11111

TABLE II:TRUTH TABLE OF THE FULL ADDER

#	Cin	A[i]	B[i]	S[i]	C_{out}	Comment
1	0	0	0	0	0	
2	**0**	0	1	1	**0**	$C_{out} = C_{in}$
3	**0**	1	0	1	**0**	$C_{out} = C_{in}$
4	0	1	1	0	1	
5	1	0	0	1	0	
6	**1**	0	1	0	**1**	$C_{out} = C_{in}$
7	**1**	1	0	0	**1**	$C_{out} = C_{in}$
8	1	1	1	1	1	

II PROPOSED SYSTEM

A new weighted random pattern design for testability is described where the shift register latches distributed throughout the chip are modified so that they can generate biased pseudo-random patterns upon demand. A two-bit code is transmitted to each weighted random pattern shift register latches to determine its specific weight. The weighted random pattern test is then divided into groups, where each group is activated with a different set of weights. The weights are dynamically adjusted during the course of the test to "go after" the remaining untested faults.

An accumulator-based 3-weight test pattern generation scheme is presented; the proposed scheme generates set of patterns with weights 0, 0.5, and 1. Since accumulators are commonly found in current VLSI chips, this scheme can be efficiently utilized to drive down the hardware of built in self test pattern generation, as well. Comparisons with previously presented schemes indicate that the proposed scheme compares favorably with respect to the required hardware.

Generally, the accumulator-based compaction technique uses an accumulator to generate a composite fault signature for a circuit under test. The error coverage for this method has been previously analyzed. We describe an alternative technique for calculating the error coverage of accumulator-based compaction using the asymmetric error model. This technique relies on the central limit theorem of statistics and can be applied to other count-based compaction schemes. The data paths of most contemporary general and special purpose processors include registers, adders and other arithmetic circuits. If these circuits are also used for built-in self-test, the extra area required for embedding testing structures can be cut down efficiently. Several schemes based on accumulators, subtracters, multipliers and shift, resisters have been proposed and analyzed in the past for parallel test response compaction, whereas some efforts have also been devoted in the bit-serial response compaction case.

MODULES

The utilization of accumulators for time compaction of the responses in built-in self test environments has been studied by various researchers. One of the well-known problems of time compactors is aliasing, i.e. the event that a series of responses containing errors result in a signature equal to that of an error-free response sequence. In this paper we propose a scheme to reduce aliasing in accumulator based compaction environments. With the proposed scheme, the aliasing probability tends to zero, as the number of the patterns of the test set increases.

We use a pseudo random generator made using Linear Feedback Shift Register (LFSR). These patterns generated using LFSR have all the desirable properties of random numbers, but are algorithmically generated by the hardware pattern generator and are therefore repeatable,

Which is essential for BIST? We no longer cover all the 2^n combinations, but a large number of test pattern sequences will still be necessary to attain sufficient fault coverage.

In general, pseudo random pattern generation requires more patterns than completely deterministic Automatic Test Pattern Generation (ATPG), but obviously, fewer than the exhaustive testing. However, it was found that the stuck-fault coverage rises in a logarithmic fashion towards hundred percentage, but at the cost of enormous numbers of random patterns. On top of it, certain circuits are random pattern resistant circuits in that they do not approach full fault coverage with an unbiased random pattern. Such circuits require extensive insertion of testability hardware or a modification of random pattern generation to 'weighted pseudo random pattern generation' in order to obtain an acceptable fault percentage. This desire to achieve higher fault coverage with shorter test lengths and therefore shorter test times led to the invention of the weighted pseudo random pattern generator.

The implementation of the weighted-pattern generation scheme is based on the full adder truth table, presented in Table. From Table Cout=Cin. Therefore, in order to transfer the carry input to the carry output, it is enough to set A[i] = NOT B[i]. The proposed scheme is based on this observation.

#	Cin	A[i]	B[i]	S[i]	C_{out}	Comment
1	0	0	0	0	0	
2	0	0	1	1	0	$C_{out}=C_{in}$
3	0	1	0	1	0	$C_{out}=C_{in}$
4	0	1	1	0	1	
5	1	0	0	1	0	
6	1	0	1	0	1	$C_{out}=C_{in}$
7	1	1	0	0	1	$C_{out}=C_{in}$
8	1	1	1	1	1	

Fig .1: TRUTH TABLE FOR FULL ADDER

ACCUMULATOR CELL

The main object of the weighted pattern generation is an accumulator cell. To implement the accumulator in the proposed weighted pattern generation scheme is based on presented in Figure.

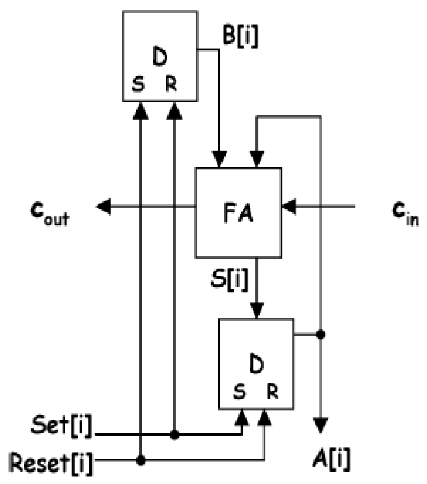

Fig .2: Accumulator cell

Which consists of a Full Adder (FA) cell and a D-type flip-flop with asynchronous set and reset inputs whose output is also driven to one of the full adder inputs. In the above figure, we assume that the set and reset are active high signals and at the same time the set and reset are used to without loss of generality. And at the time, the respective cell of another register B[i] is also occurred. For this accumulator cell, one out of three configurations can be utilized, as shown in Fig.

In Fig. we present the configuration that drives the CUT inputs. When A[i] =1 is required, So the set[i]=1 and reset[i]=0 and hence A[i]=1 and B[i]=0. Then the output is equal to 1,and Cin is equal to Cout. i.e., the Cin is transferred to the Cout.

And similarly, When A[i] =0 is required, So the set[i]=0 and reset[i]=1 and hence A[i]=0 and B[i]=1. Then the output is equal to 0, and here Cin is equal to Cout. i.e., the Cin is transferred to the Cout.

When A[i] = "-" is required, so the set[i] =0 and reset[i] =0. The D input of the flip-flop of register B is driven by either 1 or 0, depending on the value that will be added to the accumulator inputs in order to generate satisfactorily random patterns to the inputs of the CUT.

Linear Feedback Shift Registers

The LFSR are the basic building blocks of the pseudo random test pattern generators. In unbiased pseudo random testing, the outputs from the LFSR is fed directly to the CUT and thus the no. of LFSR stages required is equal to the number of inputs to the CUT. For a weighted pseudo random testing we however require much more LFSR stages than the inputs to the CUT. This is so because each weighted bit usually requires more than one equi-probable bit coming in from an LFSR stage for the generation of its weighted bit.

Now we assume that for both the unbiased and the weighted case we have the total number of LFSR shift registers required for each of the CUTs.

Pattern Generation

In the last section we see how, depending upon the no. of inputs of the CUT and the associated probabilities we can make an LFSR configuration for both, the unbiased and the weighted pseudo random testing part. We do this by writing a Verilog file with the entire configuration written in it. After that we run the Verilog simulator and get the raw patterns in a file. This reads the patterns generated from an LFSR working for weighted pseudo random testing.

The general configuration of the proposed scheme is presented. The Logic module provides the Set [n-1:0] and Reset [n-1:0] signals that drive the S and R inputs of the Register A and Register B inputs. Note that the signals that drive the S inputs of the flip-flops of Register A, also drive the R inputs of the flip-flops of Register B and vice versa.

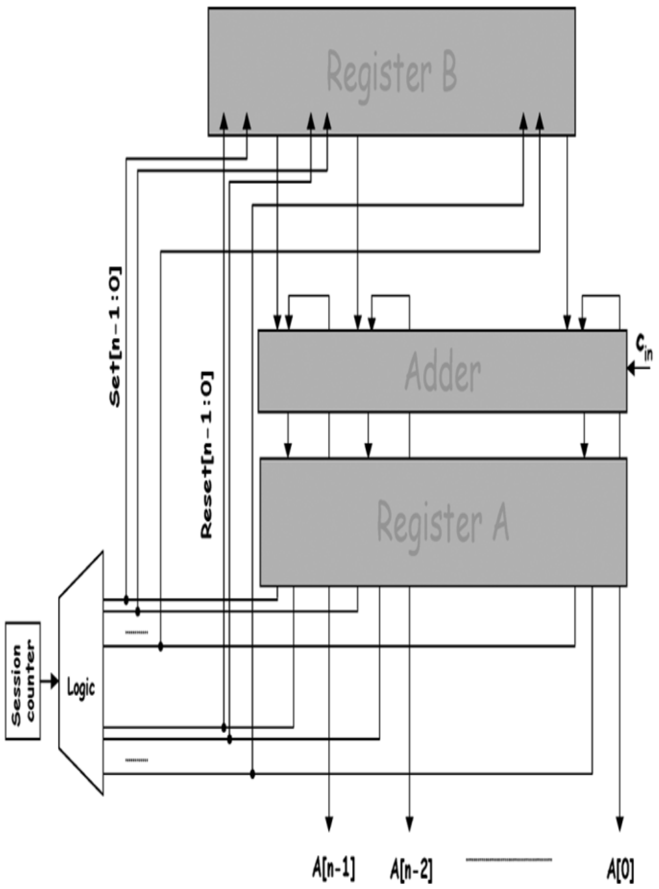

Fig .3: proposed scheme

All schemes require the application of the session counter, required to alter among the different weight sessions. The number of test patterns applied and the proposed scheme is the same, since the test application algorithms that have been invented and applied, Methods to generate weighted pseudo-random patterns for combinational circuits that can be extended to sequential circuits this method is based on the use of three weights, 0, 0.5 and 1. A weight assignment associates one of these weights with every primary input of the circuit. A preselected number of patterns N is applied under every weight assignment. A weight of 0.5 assigned to an input i by a weight assignment w implies that pseudo-random patterns are applied to input i while N test patterns are applied to the circuit; a weight of 0 assigned to input i implies that input i is held at 0 constantly for the N test patterns and a weight of 1 assigned to input i implies that input i is held at 1 constantly for the N test patterns. The weight assignments are based on a deterministic test set. Each weight assignment is obtained by intersecting a subset of deterministic test patterns. The intersection of identical values, 0 or 1, yields a weight of 0 or 1, respectively.

The intersection of different values yields an unspecified value (x), which is translated into a weight of 0.5. The intersection of test subsequences yielded weight assignments that were used in a similar way to the ones for combinational circuits. However, for sequential circuits, the intersection of a subset of test subsequences of length M results in M weight assignments that have to be used consecutively, and changed at every time unit. The need to change the weight assignment at every time unit is undesirable.

In addition, this method is not applicable when a single test sequence is given for the circuit. It can be equally well applied with both implementations. Therefore, the comparison will be performed with respect to the hardware overhead and the impact on the timing characteristics of the adder of the accumulator. Both schemes require a session counter in order to alter among the different weight sessions; the session counter consists of log 2 K bits, where K is the number of test sessions of the weighted test set. In the 3-weight pattern generation scheme proposed the scan chain is driven by the output of a linear feedback shift register (LFSR). Logic is inserted between the scan chain and the CUT inputs to fix the outputs to the required weight (0, 0.5, or 1). In order to implement the scheme [5], a scan-structure is assumed.

Furthermore, an LFSR required to feed the pseudorandom inputs to the scan inputs is implemented the number of LFSR stages is log2 n, where n is the number of scan cells, as well as a scan counter, common to all scan schemes.

Comparisons with a previously proposed accumulator-based 3-weight pattern generation technique, the hardware overhead of the proposed scheme is lower, while at the same time no redesign of the accumulator is imposed, thus resulting in reduction in test application time.

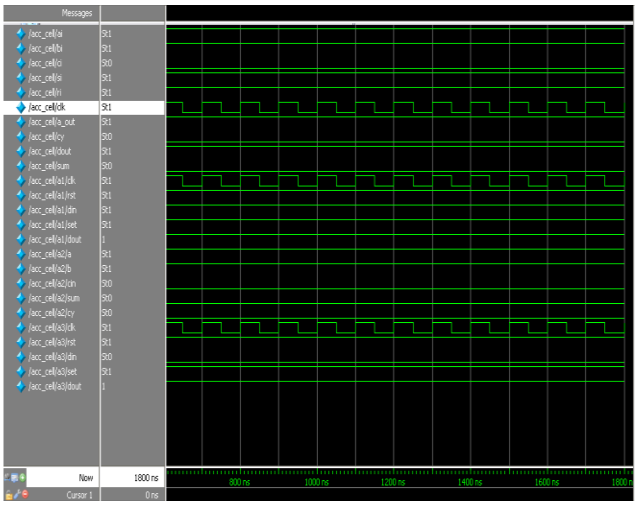
Fig.4: **Accumulator**

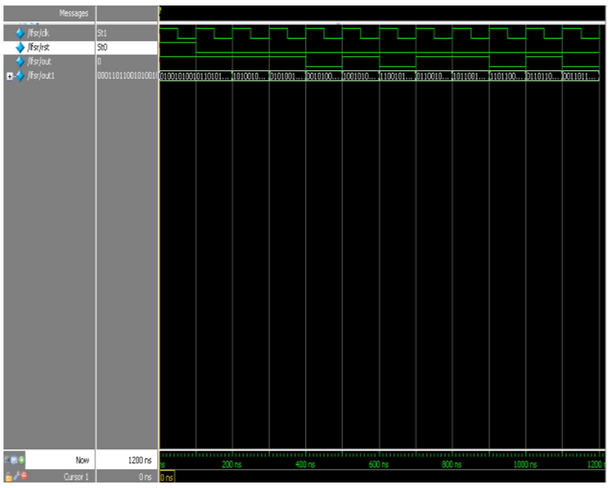
Fig.5: **Linear Feedback Shift Register**

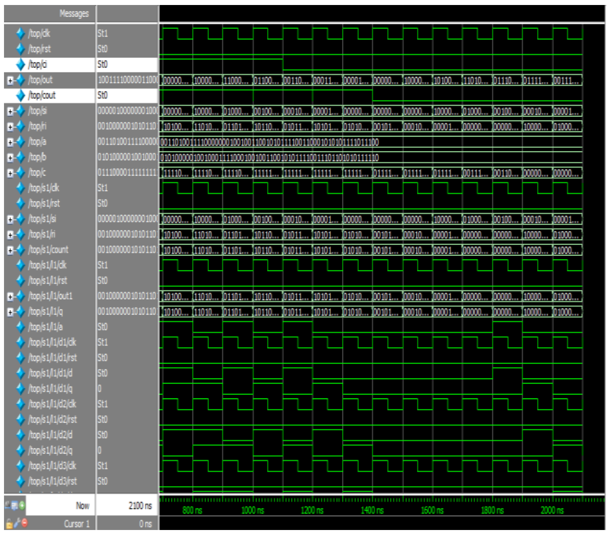
Fig.6: Topmodule for benchmark circuit

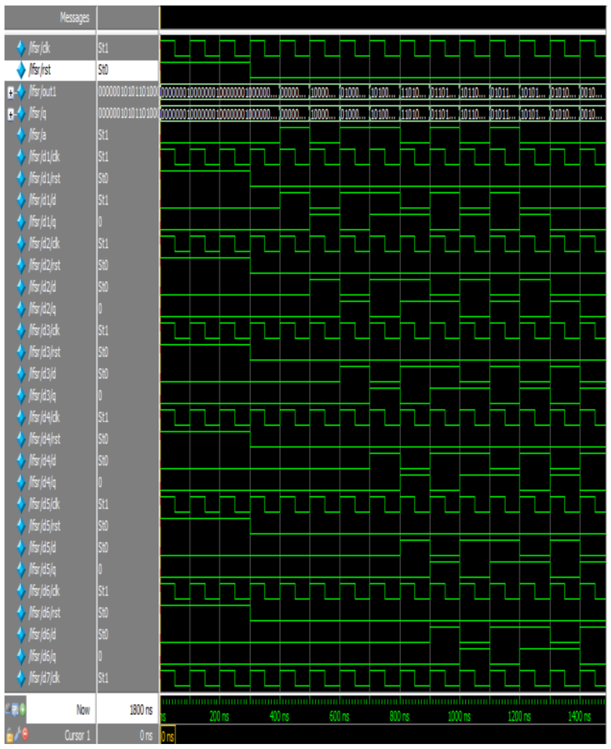
Fig.9: LFSR for benchmark circuit

Dinesh Babu al./ IJAIR *Vol. 2 Issue 7* *ISSN: 2278-7844*

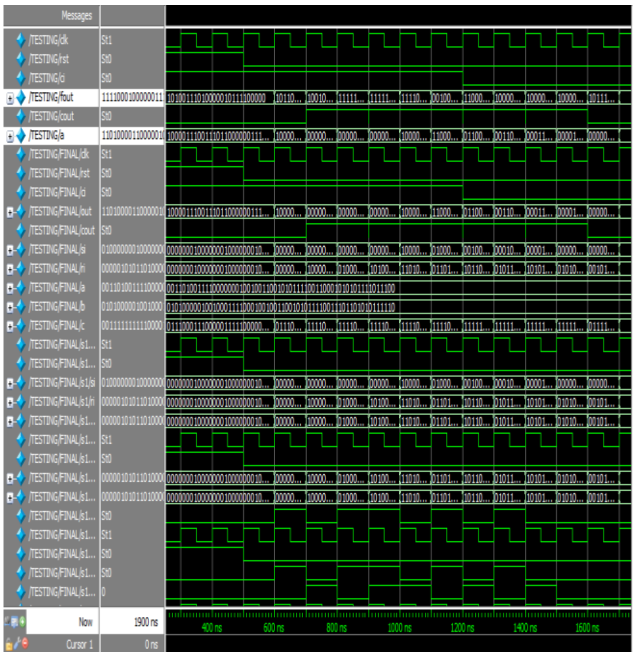

Fig.10: Benchmark circuit with topmodule

CONCLUSION

Finally, we have presented an accumulator-based 3-weight (0, 0.5, and 1) test-per-clock generation scheme, which can be utilized to efficiently generate weighted patterns without altering the structure of the adder. Comparisons with a previously proposed accumulator-based 3-weight pattern generation technique and it indicates that the hardware overhead of the proposed scheme is lower, while at the same time no redesign of the accumulator is imposed, thus resulting in reduction in test application time. Comparisons with scan based schemes show that the proposed schemes results in lower hardware overhead. Finally, comparisons with the accumulator- based scheme proposed and reveal that the proposed scheme results in significant decrease in hardware overhead.

REFERENCES

[1] P. Bardell, W. McAnney, and J. Savir, Built-In Test For VLSI: Pseudorandom Techniques. New York: Wiley, 1987.

[2] P. Hortensius, R. McLeod, W. Pries, M. Miller, and H. Card, “Cellular automata-based pseudorandom generators for built-in self test,” IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol. 8, no. 8, pp. 842–859, Aug. 1989.

[3] A. Stroele, “A self test approach using accumulators as test pattern generators,” in Proc. Int. Symp. Circuits Syst., 1995, pp. 2120–2123.

[4] H. J. Wunderlich, “Multiple distributions for biased random test patterns,” in Proc. IEEE Int. Test Conf., 1988, pp. 236–244.

[5] I. Pomeranz and S. M. Reddy, “3 weight pseudo-random test generation based on a deterministic test set for combinational and sequential circuits,” IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol.12, no. 7, pp. 1050–1058, Jul. 1993.

[6] K. Radecka, J. Rajski, and J. Tyszer, “Arithmetic built-in self-test for DSP cores,” IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol. 16, no. 11, pp. 1358–1369, Nov. 1997.

[7] J. Rajski and J. Tyszer, Arithmetic Built-In Self Test For Embedded Systems. Upper Saddle River, NJ: Prentice Hall PTR, 1998.

[8] S. Wang, “Low hardware overhead scan based 3-weight weighted random BIST,” in Proc. IEEE Int. Test Conf., 2001, pp. 868–877.

[9] S. Zhang, S. C. Seth, and B. B. Bhattacharya, “Efficient test compaction for pseudo-random testing,” in Proc. 14th Asian Test Symp., 2005, pp. 337–342.

[10] J. Savir, “Distributed generation of weighted random patterns,” IEEE Trans. Comput., vol. 48, no. 12, pp. 1364–1368, Dec. 1999.

[11] I. Voyiatzis, D. Gizopoulos, and A. Paschalis, “Accumulator-based weighted pattern generation,” presented at the IEEE Int. Line Test Symp., Saint Raphael, French Riviera, France, Jul. 2005.

[12] F. Brglez and H. Fujiwara, “A neutral netlist of 10 combinational benchmarks circuits and a target translator in FORTRAN,” presented at the Int. Symp. Circuits Syst., Kyoto, Japan, 1985.