A Low-Cost Optimal Time SIC Pair Generator

I. Voyiatzis[1], H. Antonopoulou[2], C. Efstathiou[1]
[1]*Department of Informatics, Technological Educational Institute of Athens, Greece*
[2] *Technological Educational Institute of Patras, Computer Technology Institute, Patras, Greece*
voyageri@otenet.gr

***Abstract* - The application of Single Input Change (SIC) pairs of test patterns is very efficient for sequential, i.e. stuck-open and delay fault testing. In this paper a novel implementation for the application of SIC pairs is presented. The presented generator is optimal in time, in the sense that it generates the n-bit SIC pairs in time n×2n, i.e. equal to the theoretical minimum. Comparisons with the schemes that have been proposed in the open literature which generate SIC pairs in optimal time reveal that the proposed scheme requires less hardware overhead**

***Keywords* - Built-In Self Test, Two-Pattern Testing, Delay Fault Testing, Stuck-Open Testing**

I. Introduction

While every VLSI design project has its own unique set of goals, there is a fundamental need for reliability in the finished product. Built-In Self Test (BIST) [1] constitutes an attractive and practical solution. Advantages of BIST include the possibility of performing at-speed testing, very high fault coverage, elimination of test generation effort and less reliance on expensive external testing equipment for applying and monitoring test patterns. Therefore BIST reduces the cost of testing. With the increasing complexity of today's VLSI devices (with millions of gates) BIST schemes for embedded modules are increasingly becoming a necessity.

Despite of the fact that exhaustive single-pattern testing provides for 100% fault coverage of detectable single and multiple stuck-at faults without the need for fault simulation or deterministic test pattern generation, it is widely known that a large class of physical defects can not be modeled as stuck-at faults. For example, a transistor stuck-open fault in a CMOS circuit can convert a combinational Circuit Under Test (CUT) into a sequential one [2] while a delay fault (although it does not affect the steady-state operation) may cause circuit malfunction at clock speed [3]. Detection of such faults requires two-pattern tests.

In the literature, two largely known types of two-pattern tests have been investigated, Multiple Input Change (MIC) and Single Input Change (SIC) pairs. SIC pairs are pairs of patterns in which the first pattern differs from the second one in exactly one bit. The utilization of SIC pairs for the detection of stuck-open and delay faults holds some very interesting properties and has been studied by a number of researchers both theoretically [11] and experimentally [29], [31]-[38]. In the theoretical field, Smith [11] proved that SIC tests are sufficient to detect all robustly detectable path delay faults. In the experimental field, Wang and Gupta [6] proved that SIC pairs provide higher pseudorandom robust path delay fault coverage than MIC pairs. In other words, if a certain number of pairs is applied to the inputs of a Circuit Under Test (CUT), if the pairs are SIC, the achieved fault coverage will be higher than the case in which the pairs are MIC. Gizdarski [40] utilized SIC sequences in order to test delay faults in the address decoders of RAM memories. The above-referenced results, as well as a number of related works [31-38] indicate that the utilization of SIC pairs for testing delay and stuck-open faults compares favorably to the utilization of MIC pairs, since it results in higher fault coverage with fewer test vectors.

In this paper a novel technique is presented for the generation of SIC pairs of patterns. The number of cycles required to generate the SIC pairs is n×2n, i.e. equal to the theoretical minimum. Comparisons with schemes proposed previously for the application of SIC pairs in optimal time indicate that the proposed scheme requires less hardware overhead.

The paper is organized as follows. In Section 2 the proposed scheme is introduced. In Section III the hardware implementation is presented. In Section IV the techniques presented in the literature for the generation of SIC pairs in optimal time are compared. Finally, in Section V we conclude the paper.

II. Implementation of the Proposed Scheme

Definition 1. We define by $G_n = (g_{n-1}, g_{n-2}, \ldots, \ldots, g_1, g_0)$ the 2^n-row by n column matrix that is the output of a binary-reflected gray code.

For example, for n=3, $G_3 = (g_2, g_1, g_0)$ is the 8-row 3 column matrix presented in the sequel.

000
001
011
010
110
111
101
100

Definition 2: We define by $G^i = T^i(G)$, $1 \leq i < n$ the 2^n-row by n column matrix that is generated from G by cyclically shifting the columns one position to the right and inverting the high- and low-order columns.

For example, for G= (g_2, g_1, g_0), $G^1 = T^1(G) = (g_0', g_2, g_1')$, $G^2 = T^2(G) = T^1(T^1(G)) = (g_1, g_0', g_2')$ and $G^3 = T^3(G) = T^1(T^2(G)) = T^1(T^1(T^1(G))) = (g_2\ g_1\ g_0) = G$. In the following table, we present the matrices G, G^1 and G^2 for n=3.

G = (g2 g1 g0)	**$G^1 = T^1(G)$ = (g_0' g_2 g_1')**	**$G^2 = T^2(G)$ = (g1 g0' g2')**
000	101	011
001	001	001
011	000	101
010	100	111
110	110	110
111	010	100
101	011	000
100	111	010

It is trivial to show that for any value of n, $G_n^n = T^n(G_n) = G_n$.

Hayes [41] proposed a procedure to construct all SIC pairs within $n \times 2^n$ cycles by applying the n sequences G_n, G_n^1, G_n^2, … G_n^{n-1}, and presented an intuitive proof for the correctness of the construction. The proposed generator for the generation of the SIC pairs in optimal time is presented in Figure 1.

The module depicted in Figure 1 comprises an n-stage gray counter (an n-stage counter and n-1 2-input XOR gates), an n-stage barrel shifter, a k-stage counter, a k-input OR gate, a k-to-n decoder with enable and a series of 2-input XOR gates. It operates as follows. Initially, both counters are reset to 0. The n-stage counter starts increasing, hence the sequence G_n is generates at the A[n-1:0] outputs of the generator. When the n-stage counter reaches 2^n-1, the k-stage counter increases to 1. Hence, the outputs of the n-stage counter are shifted one position to the right by the barrel shifter, the output of the OR gate is 1 and the decoder output is 00…01. Therefore, the sequence $G_1 = (g_0'\ g_{n-1}\ g_{n-2}\ \ldots\ g_2\ g_1')$ is generated. When this completes, the k-stage counter is increased again, the shifter shifts the outputs of the gray counter two positions to the right, the output of the OR gate becomes 1 again and the decoder output becomes 00…010; therefore, the sequence $G_2 = (g_1\ g_0'\ g_{n-1}\ g_{n-2}\ \ldots\ g_2')$ is generated and so on.

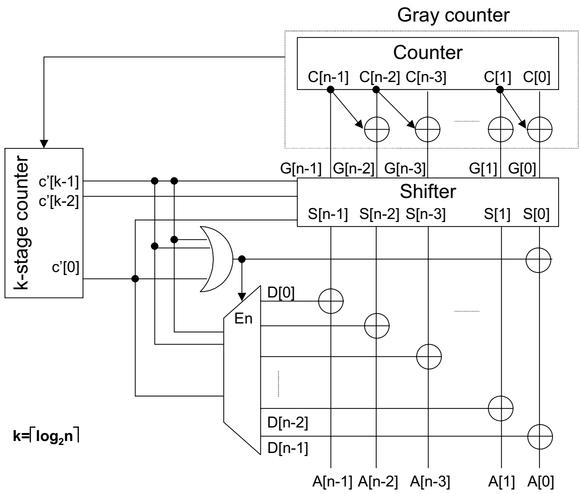
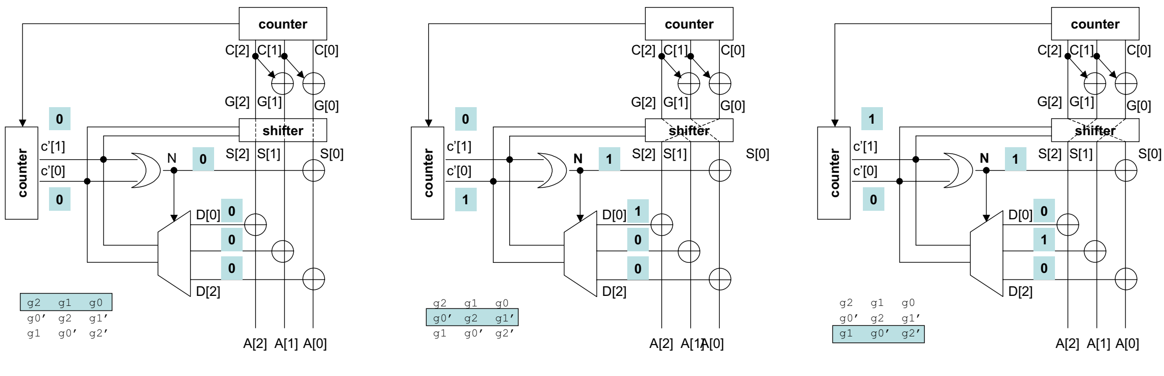


Fig. 1. The proposed generator

In order to exemplify the operation of the proposed generator, in Figure 2 we present the operation for the case n=3. During the first phase, Figure 2 (a), the $G_3 = (g_2\ g_1\ g_0)$ sequence is applied to the A[2:0] outputs. During the second phase, Figure 2 (b), the sequence $G_3^1 = (g_0'\ g_2\ g_1')$ is applied to the A[2:0] outputs; finally, during the third phase, the sequence $G_3^2 = (g_1, g_0'\ g_2')$ is applied.



a b c

Fig. 2. Operation of the proposed generator for n=3

III. CALCULATION OF THE HARDWARE OVERHEAD OF THE PROPOSED SCHEME

In the application of the proposed scheme, implementing the Gray generator requires an n-bit counter accompanying (n-1) XOR gates; also, the n-bit barrel shifter is required ($n \times \log_2 n$ flip flops) the k-stage counter ($k=\log_2 n$), the k-to-n decoder with enable, a k-input OR gate, and (n+1) additional XOR gates are required. To calculate the hardware overhead of the k-to-n decoder, we follow a reasoning similar to that used in [27].

A k-to-K decoder can be implemented as follows. Let $k_1=\lfloor \frac{k}{2} \rfloor$ and $k_2=\lceil \frac{k}{2} \rceil$. Then $k_1+k_2=k$. A k-to-K decoder ($K=2^k$) with enable input can be implemented using two subdecoders with enable (k_1-to-K_1) and (k_2-to-K_2) and K 2-input NOR gates. The first k_1-to-K_1 subdecoder is denoted by D_a; its inputs are denoted by d_{a0} to d_{ak1-1} and its outputs are denoted by D_{a0} to D_{aK1-1}; the second subdecoder is denoted by D_b; its inputs are denoted by d_{b0} to d_{bK2-1}; its outputs are denoted by D_{b0} to D_{bK2-1}. All outputs of the two subdecoders are inverted using K_1+K_2 inverters. Each one of the K gates takes two inputs: the first is an output of the first decoder; the second is an output of the second decoder as follows. $D_0=\overline{D_{a0}+D_{b0}}$; $D_1=\overline{D_{a0}+D_{b1}}$, ..., $D_{K-1}=\overline{D_{aK1-1}+D_{bK2-1}}$. For example, in Figure 3 we present a 3-to-8 decoder using the above-mentioned procedure.

For the proposed scheme, only n out of 2^k outputs are implemented ($n \le 2^k$). In Table I the Hardware Overhead (in transistors) for various values of n, the inputs of the CUT is presented.

TABLE I
Patterns generated by the module in Figure 1

c'[1:0]	C[2:0]	G[2:0]	S[2:0]	N	D[0:2]	A[2:0]
00	000	000	000	0	000	000
	001	001	001			001
	010	011	011			011
	011	010	010			010
	100	110	110			110
	101	111	111			111
	110	101	101			101
	111	100	100			100
01	000	000	000	1	100	101
	001	001	100			001
	010	011	101			000
	011	010	001			100
	100	110	011			110
	101	111	111			010
	110	101	110			011
	111	100	010			111
10	000	000	000	1	010	011
	001	001	010			001
	010	011	110			101
	011	010	100			111
	100	110	101			110
	101	111	111			100
	110	101	011			000
	111	100	001			010
00	000	000	000	0	000	000

In Table I, in the first column we present the value of n and in the second column the value of k=log2n; in the third and fourth columns we present the k1 and k2 values such that k1+k2=k; in the seventh and eighth columns the hardware overhead of the two sub-decoders is presented; in the ninth column the overhead of the n 2-input gates is presented. In the tenth column, the hardware overhead of the decoder for each value of n is presented. This value will be considered for the calculation of the hardware overhead of the proposed scheme. For the calculations, an m-input NAND/NOR gate is considered to have $2m$ transistors and an m-input AND $2m+2$ transistors [23]. The hardware overheads of the 2x4, 3x8 and 4x16 decoders are 26, 66, and 116 transistors respectively.

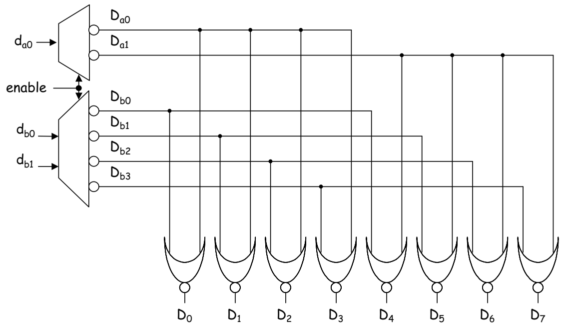


Fig. 3. Implementation of 3-to-8 decoder utilizing smaller decoders and 2-input gates

TABLE II
Calculation of k-to-n decoder ($n \le 2^k$) with enable hardware overhead (in transistors)

n	k	Dec₁	Dec₂	Dec₁ H/W	Dec₂ H/W	n 2-input gates	Dec H/W
12	4	2x4	2x4			48	100
16					26	64	116
20	5		3x8			80	172
24						96	188
28						112	204
32				26		128	220
36	6	3x8				144	276
40						160	292
44						176	308
48						192	324
52						208	340
56						224	356
60						240	372
64					66	256	388
68	7		4x16			272	404
72						288	470
76						304	486
80						320	502
84						336	518
88						352	534
92						368	550
96						384	566
100						400	582
104						416	598
108						432	614
112						448	630
116						464	646
120						480	662
124						496	678
128				66	116	512	694

IV. COMPARISONS

In this section, the proposed scheme will be compared with the techniques proposed hitherto for the generation of SIC pairs in optimal time, i.e. in exactly n×2n clock cycles [5], [37], [40] in terms of the required hardware overhead.

In PEAT [5], an n-stage NFSR, an n-stage *shift register* and an n-stage *shift register with flip capability* are utilized to generate the SIC pairs within $(n+1)\times 2^n$ clock cycles. To implement the technique, the NFSR and *n scan flip-flops with flip capability* are implemented. Furthermore, the *n* flip-flops of the existing register are substituted by *scan flip-flops*.

In [37], Das *et al* presented an optimal solution to the problem of generating SIC pairs, in the sense that the pairs are generated within time equal to the theoretical minimum, i.e. $n\times 2^n+1$. However, the hardware overhead of [37] is rather high, thus the value of the scheme lies mainly on its high theoretical significance. The hardware overhead of the scheme is, according to [37], 3n+2 flip flops, n XOR gates (2-input), ($2n$-1) OR gates (2-input), n+1 AND gates (2-input) and 1 NOT gate.

Gizdarksi [40] utilized the algorithm proposed by Hayes in order to generate the SIC pairs to the inputs of the address decoder of a RAM. Gizdarski utilized two sequences, called TS1 and TS2 in [40]; TS2 is utilized in order to detect additional faults in the address decoder (and its generation is more complicated and hardware intensive). Hence the generator for the TS1 sequence is considered for our comparisons. The required hardware includes control logic, an n-bit binary counter, an n-bit register, n 2-input gates, and $n\times\log_2 n$ 2-to-1 multiplexers in a barrel shifter. Since no information is provided in [40] for the control logic, we shall not take it into account in our comparisons.

For the comparisons, the following are taken into account [23]. A 2-input NAND/NOR gate requires 4 transistors; a 2-input AND requires 6 transistors and a 2-input XOR gate can be implemented by 4 CMOS transistors [42]. The memory elements used are considered to have set/reset capability. Thereby, the flip-flop requires 26 transistors, the *scan flip-flop* requires 34 transistors and the *scan flip-flop with flip capability* [5] requires 46 transistors.

In Table III we present, for each one of the optimal SIC pair generation techniques (first column) the formulas used for the calculation of the hardware overhead (second column) and the hardware overhead (in transistors, third column). In Figure 4 we present, for various values of n, the ratio of the hardware overhead (in transistors) over n, the number of CUT inputs. From Figure 4 it can be concluded that the proposed scheme presents the least hardware overhead of the schemes that have been proposed in the open literature.

V. CONCLUSION

In this paper a novel generator for Single Input Change two-pattern tests has been presented. The number of cycles required to generate the SIC pairs is $n\times 2^n$, i.e. equal to the optimal (minimum) time required. Comparisons with the techniques that have been proposed in the literature for the generation of SIC pairs in optimal time revealed that the proposed scheme requires less hardware overhead.

TABLE III
Optimal-time SIC pair generation techniques: Comparison

	Hardware Overhead	
Technique	**Modules**	**Transistors**
Peat [5]	$n\times(\mathrm{DFF}+\mathrm{NOR})+n\times \mathrm{DFF}_{scanwithflip}+n\times \mathrm{DFF}_{scan}$	$110\times n$
Gizdarski [40]	Control + n×DFF + n×DFF + $n\times \mathrm{AND}_2$ + $n\times\log_2 n\times \mathrm{MUX}_{21}$	n×(58+log2n)
DAS [37]	(2n+2)×DFF + n×XOR + $(2n-1)\times \mathrm{OR}_2$ + $(n+1)\times \mathrm{AND}_2$ + NOT	84×n+40
Proposed	n×DFF + (n-1)×XOR + $\log_2 n\times$DFF + $n\times\log_2 n\times \mathrm{MUX}_{21}$ + log2n-to-n Dec + (n+1)×XOR + $\log_2 n$-input OR	$n\times(38+6\times\log_2 n)+20\times\log_2 n$

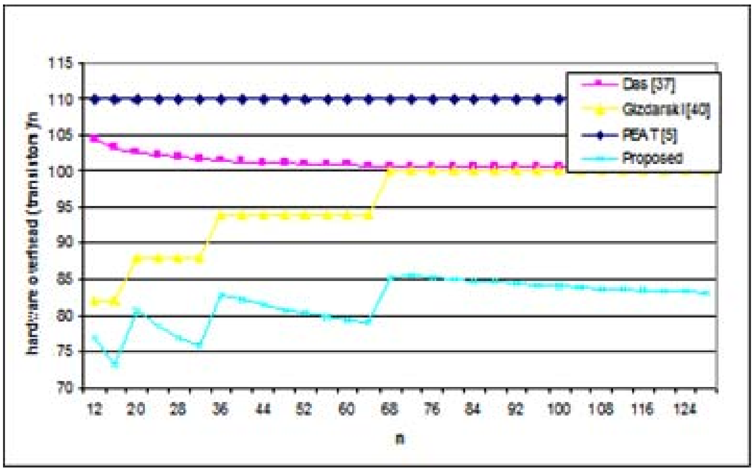


Fig. 4. Optimal time SIC pair generation schemes: Comparison

REFERENCES

[1] Abramovici M., Breuer M., Freidman A., *"Digital Systems Testing and Testable Design"*, Computer Science Press, 1990.

[2] Wadsack R. L., *"Fault Modeling and logic simulation of CMOS and MOS integrated circuits"*, Bell Syst. Tech. J., vol. 57, no. 5, pp. 1449-1474, May-June 1987.

[3] Woods M.H., "*MOS VSI Reliability and Yield Trends",* Proceedings of the IEEE, vol. 74, no. 12, Dec. 1986, pp. 1715-1729.

[4] C. Dufaza, Y. Zorian, On the Generation of Pseudo-Deterministic Two-Pattern Test Sequences with LFSRs, Proceedings of the European Design and Test Conference, Paris, France, March 17-20 1997, pp. 69-76.

[5] Craig G., Kime Ch., *"Pseudo-Exhaustive Adjacency Testing: A BIST Approach for Stuck-open Faults"*, IEEE International Conference, pp. 126-137, 1985.

[6] Wang W., Gupta S., *"Weighted Random Robust Path Delay Testing of Synthesized Multilevel Circuits"*, IEEE VLSI Test Symposium, pp. 291-297, 1994.

[7] Girard P., Landrault C., Moreda V., Pravossoudovitch S., "*An Optimized BIST Test Pattern Generator for Delay Testing*", in. Proc. VLSI Test Symposium, 1997, pp. 94-100.

[8] Voyiatzis I., Kranitis N., Gizopoulos D., Paschalis A., Halatsis C., "An Accumulator-based Built-In Self-Test Generator for Robustly Detectable Sequential Fault Testing", IEE Proceedings, Computers and Digital Techniques, vol. 151, no. 6, pp. 466-472, November 2004.

[9] S. Nandi, B. Vamsi, S. Chakraborty, P.Pal Chaudhuri, *Cellular Automata as a BIST structure for testing CMOS circuits*, IEE Proc.-Com. Digit. Tech., Vol. 141, No. 1, pp. 41-47, January 1994.

[10] Voyiatzis I., Paschalis A., Nikolos D., Halatsis C., *"An Efficient Built-In Self Test Method for Robust Path Delay Fault Testing"*, Journal of Electronic Testing: Theory and Applications, pp. 219-222, June 1996.

[11] Smith G. L., *"Model for Delay Faults based Upon Paths"*, IEEE International Test Conference, pp.309-314, 1984.

[12] E. Blokken, H. de Keulenaer, F. Catthoor, H. J. de Man, *A flexible module library for custom DSP applications in a Multiprocessor Environment*, IEEE J. Solid-State Circuits, vol. 25, no. 3, pp. 720-729, June 1990.

[13] R.J. Higgins, *Digital Signal Processing in VLSI*, Englewood Cliffs, NJ: Prentice Hall, 1990.

[14] Rajski J., Tyszer J., *"Accumulator-Based Compaction of Test Responses",* IEEE Transactions on Computers, vol. 42, no. 6, pp.643-650, June 1993.

[15] Rajski J., Tyszer J., *"Test Responses Compaction in Accumulators with Rotate Carry adders",* IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 12, no. 4, pp. 531-539, April 1993.

[16] A.P. Stroele, "*Test Response Compaction Using Arithmetic Functions*", Proceedings of the 14th VLSI Test Symposium, pp. 380-386, 1996.

[17] J. Rasksi and J. Tyszer, *Arithmetic Built-In Self Test for Embedded Systems*, Prentice Hall PTR, Upper Saddle River, New Jersey, 1998.

[18] Gupta S., Rajski J., Tyszer J., *"Arithmetic Additive Generators of Pseudo-Exhaustive Test Patterns",* IEEE Transactions on Computers, vol 45, no.8, pp.939-949, August, 1996.

[19] A.P. Stroele, "*BIST Pattern Generators Using Addition and Subtraction Operations*", Jetta, 11, pp. 69-80, 1997.

[20] K. Radecka, J. Rajski, J. Tyszer, "*Arithmetic Built-In Self Test for DSP Cores", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*", vol. 16, no. 11, pp. 1358-1369, November 1997.

[21] I. Voyiatzis, A. Paschalis, D. Nikolos, and C. Halatsis. "*Accumulator-Based BIST Approach for Stuck-Open and Delay Testing*", Proc. of the European Design and Test Conference 1995, March 1995, pp. 431-435.

[22] Mano M., "*Digital Design*", Prentice Hall Int., 3rd ed., 2002.

[23] Weste N., Eshraghian K., "*Principles of CMOS VLSI Design: A Systems Perspective*", Addison Wesley Company 1985.

[24] C.W. Starke, *Built-In Test for CMOS Circuits*, Proceedings of the IEEE Int. Test Conf., pp. 309-314, Oct. 1984.

[25] A. Vuksic, K. Fuchs, *A New BIST Approach For Delay Fault Testing*, Proceedings of the European Design and Test Conference, pp. 284-288, March 1994.

[26] Ch. Chen, S. K. Gupta, "*BIST Test Pattern Generators for Stuck-open and Delay Testing",* Proceedings of the European Design and Test Conference, pp. 289-296, March 1994.

[27] Voyiatzis I., Th. Haniotakis, C. Halatsis, "A Novel Algorithm for the Generation of SIC Pairs and its Implementation in a BIST environment", IEE Proc. Circuits, Devices & Systems, vol. 153, Issue 5, October 2006, pp. 427-432.

[28] J. Rajski, J. Tyszer, "*Recursive Pseudoexhaustive Test Pattern Generation*", IEEE Transactions on Computers, vol. 42, no. 12, December 1993, pp.1517-1521.

[29] Girard P., Landrault C., Pravossoudovitch S., Virazel A., "*Comparison between random and pseudorandom generation for BIST of delay, stuck-at and bridging faults*", IEEE On-Line Testing workshop, pp. 121-126, 2000.

[30] Voyiatzis I., Paschalis A., Nikolos D., Halatsis C., "*Exhaustive and Pseudoexhaustive Built-In Two-Pattern Generation for Datapaths*", IEEE International On-Line Test Workshop, 1998.

[31] David R., Girard P., Landrault C., Pravossoudovitch S., Virazel A., "*On using Efficient Test sequences for BIST*", VLSI Test Symposium 2002.

[32] Virazel A., David R., Girard P., Landrault C., Pravossoudovitch S., "*Delay Fault Testing: Choosing between Random SIC and Random MIC sequences*", 2000, IEEE European Test Workshop, pp. 9-14.

[33] Rahaman H, Das D., Bhattacharya B., "*Transition count based BIST for Detecting Multiple Stuck-open Faults in CMOS circuits*", The Second IEEE Asia Pacific Conference on ASICs, Aug. 28-30, 2000.

[34] Crepaux-Motte, Jacomino M., David R., "*An algebraic Method for Delay Testing*", 14th VLSI Test Symposium, 1996, pp. 308-315.
[35] Gharaybeh M., Bushnell M., Agrawal V., "*Parallel Concurrent Path Delay Fault Simulation Using Single-Input Change Patterns*", 9th IEEE International Conference on VLSI Design, pp.426-431, Jan. 1996.
[36] Gharaybeh M., Bushnell M., Agrawal V., "*A parallel-vector Concurrent fault Simulator and Generation of Single-Input-Change Tests for Path-delay Faults"*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 17, no. 9, September 1998.
[37] Das D., Chaudhuri I., Bhattacharya B., "*Design of an Optimal Test pattern Generator for Built-In Self Testing of Path Delay Faults*", Proc. VLSI-97, pp. 205-210, 1997.
[38] S. Lu, M. Lu, "*Testing Iterative Logic Arrays for Delay Faults with a constant number of patterns*", 2002 Int'l Symposium on Electronic Materials and Packaging, pp.492-498.
[39] Brglez F., Bryan D., Kozminski K., "*Combinational Profiles of Sequential Benchmark Circuits*", in Proceedings of the International Symposium on Circuits and Systems (ISCAS), pp. 1229-1234, IEEE, 1989.
[40] Gizdarski E., "*Detection of Delay Faults in Memory Address Decoders*", Journal of Electronic Testing, Volume 16, Number 4, August 2000, pp. 381 – 387.
[41] J. P. Hayes, "*Testing Memories for Single-Cell Pattern Sensitive Faults*", IEEE Trans. on Computers, vol. C-29, no. 3, March 1980.
[42] H. T. Bui, A. K. Al-Sheraidah, Y. Wang, "New 4-transistor XOR and XNOR designs", in Proceedings of the Second IEEE Asia Pacific Conference on ASICs, 2000, pp. 25-28.