TEST PATTERN GENERATOR FOR HYBRID TESTING OF COMBINATIONAL CIRCUITS

Davide De Caro (1), Nicola Mazzocca (2), Ettore Napoli (1), Giacinto P. Saggese (1), Antonio G.M. Strollo (1)

(1) Dept. of Electronics and Telecommunications Engineering, Univ. of Naples – "Federico II",
Via Claudio 21, 80125 Napoli, Italy - Fax: +39 081 5934448 - Email: saggese@unina.it
(2) Dept. Of Information Engineering, Second Univ. of Naples,
Via Roma 29, 81031 Aversa (CE), Italy - Fax: +39 081 5037042

Abstract

A novel test pattern generator for Built-In Self-Test technique of testable combinational circuits is proposed. The presented solution is based on a hybrid testing reconfigurable test pattern generator, that uses a shift register reacted through two different networks: it firstly reproduces a pseudo-random sequence of test patterns using a linear feedback combinational network and then reproduces deterministic precalculated test patterns, useful to detect hard faults, using a non-linear feedback combinational network.

For the proposed approach, a synthesis tool (based on state space heuristic search and the "selfish gene" genetic algorithm) able to determine test pattern generator for a given test set, is also proposed. Experiments to evaluate synthesis time and the test sequence length are conducted on well-known ISCAS'85 circuits. Comparison with previous techniques shows the effectiveness of proposed solution.

1. INTRODUCTION

Testing an electronic system aims at detecting faults introduced during the manufacturing process [1]. The test is usually performed by applying a test set, composed of test vectors, generated by an Automatic Test Pattern Generator (ATPG), to the Unit Under Test (UUT) and checking its response to the expected one.

In a Built-In Self Test (BIST) technique, the generation and application of test vectors, as well as the resulting output analysis, are embedded in the system. Basic BIST scheme is shown in Fig. 1. Four blocks are added to the UUT: the Test Pattern Generator (TPG) generates the test patterns to be applied to the UUT, the Output Response Analyzer (ORA) compares the UUT response to the expected one and generates the Good/Faulty signal; a multiplexer switches UUT inputs from the Primary Inputs (PI) to the internal TPG generated ones; the BIST Controller, activated in test mode through the external Normal/Test input, manages the whole system. Key issue in BIST design are better Test Pattern Generators (TPG) and the reduction in test overhead.

So far, most solutions for embedded TPG are based on the autonomous Linear Feedback Shift Register (LFSR). These structures can be exploited as exhaustive generators, pseudo-random generators [1-5], and to reproduce deterministic sequences of test patterns [1,3,4]. Linearity of LFSR, whose reaction net is composed by only XOR gates, simplifies synthesis phase, so TPG design is brought back to the design of finite state system.

The study of shift registers reacted by a non linear (NLFSR) net for deterministic TPG, has been prevented by the difficult research of effective synthesis procedures [6].

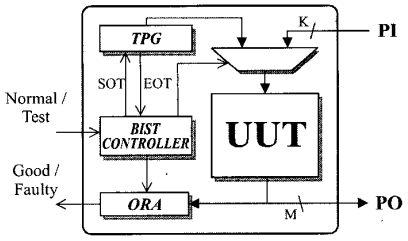

Fig.1: The BIST approach

In this paper a novel TPG, named Hybrid TPG in the following, for Built-In Self-Test of testable combinational circuits, is proposed. Hybrid TPG is a structure that, in a first phase allows the pseudo-random generation of vectors, and in a second phase reproduces vectors devoted to the detection of hard faults (the faults resistant to random testing). A synthesis tool based on heuristic search and the "selfish gene" genetic algorithm, able to determine test pattern generator for a given test set, is also presented.

2. HYBRID TPG

It has been observed that, using a random/pseudo-exhaustive testing, the fault coverage as a function of the number of test patterns tends to saturate. A knee point exists, in which a small growth of the fault coverage requires a significant increase of the number of test patterns [1,2]. This is due to those faults (named "hard faults") that can be revealed by few testing vectors and that are, therefore, hard to detect with a random/pseudo-exhaustive procedure (as example a stuck-at-0 fault on the output of an AND gate with N inputs).

An interesting solution is to complete random testing with appropriate vectors chosen to detect hard faults. Main advantage of this two-step technique (sometimes referred as hybrid, or mixed mode testing) is that structures able to generate pseudo-random vectors are particularly compact (i.e. LFSR) and that only a reduced number of deterministic test vectors must be generated. Anyway, area overhead due to the need of two separate structures, for random testing and deterministic testing, is a disadvantage that should be carefully considered.

The Hybrid TPG, see Fig. 2, is composed of a right shift register (RSR) or a left shift register (LSR) on K bit, exploited for both phases of testing. The output of the shift register is applied to the combinational unit under test (UUT) and is used by the two nets with K inputs and one output (selected through the multiplexer in different phases) to determine the next bit serially introduced in the shift register. In the first phase the shift register is reacted with a linear network to generate pseudo-random and pseudo-exhaustive test vectors (matter concerning the

0-7803-7057-0/01/$10.00 ©2001 IEEE.

determination of the seed and the reaction net have been deeply studied in the past [1,7,8]). In the second phase a multiplexer selects the non-linear reaction network, in order to complete the test set with few precalculated vectors needed to reach higher fault coverage. The block named as Controller in Figure 2 is reset by *SOT* and selects the linear net through the multiplexer until all vectors of the pseudo-random sequence are reproduced; afterwards, Controller selects the non-linear net that generates deterministic test vectors T which can detect all the hard faults not revealed through random testing. When the evolution of the reacted shift register has covered all the vectors of T, *EOT* signals the end of the testing procedure.

3. SYNTHESIS PROCEDURES

3.1 Problem formalization

In order to choose the appropriate linear feedback net that configures the Hybrid TPG as a LFSR, it is possible to refer to previous works regarding the LFSR as a pseudo-random generator [1-5,7,8]. So the problem is the choice of the non-linear feedback net able to generate, reacting the shift register, the set of test patterns for the hard faults.

Let $T_S = \{\underline{x}_1 , \underline{x}_2 , \dots , \underline{x}_N\}$ be the test set for a combinational UUT, derived by an ATPG tool which accepts circuit description, a list of potential faults (the hard faults in this case) and generates an appropriate test set.

The test set T_S that detects the hard faults often presents many don't cares and therefore an effective synthesis procedure should try to exploit the not specified bits, fixing them in order to minimize the length of the testing sequence.

If the UUT is combinational, test patterns do not need to be applied in a given order because circuit response does not depend on previous inputs. The same happens for sequential UUTs than can be considered combinational if scan-path techniques are used.

Proposed structure is based on the possibility of reordering test vectors and on the addition between vectors of T_S of intermediary vectors not belonging to T_S. In this way a sequence of vectors $T = (\underline{t}_1 , \underline{t}_2 , \dots , \underline{t}_M)$ including T_S, which can be generated by a structure of the type of Fig.2 can be found. Note that the shift register can not assume the same state twice, otherwise it falls in a loop, and hence, in a sequence T, every vector must be different from each other.

Problem formulation is: given test set $T_S = \{\underline{x}_1 , \underline{x}_2 , \dots , \underline{x}_N\}$, formed by N vectors K bit long, each different from each other, fixed a shifting direction (RSR or LSR), let us find a sequence $T = (\underline{t}_1 , \underline{t}_2 , \dots , \underline{t}_M)$ so that:

1) T includes T_S: $T \supseteq T_S$
2) all the couples of consecutive vectors of T $(\underline{t}_i , \underline{t}_{i+1})$, where $i \in \{1,\dots, M-1\}$, are results of right (respectively left) shifting, in other words $\underline{t}_{i+1}$ must be obtained from $\underline{t}_i$ by means of one left (respectively right) shift and introducing one bit, α:
   RSR: $\forall j \in \{2, \dots, K\}$ $\underline{t}_{i+1, j} = \underline{t}_{i, j-1}$, where $\underline{t}_{i+1, 1} = \alpha$
   LSR: $\forall j \in \{1, \dots, K-1\}$ $\underline{t}_{i+1, j} = \underline{t}_{i, j+1}$, where $\underline{t}_{i+1, K} = \alpha$
3) T vectors must be different from each other: $\underline{t}_i \neq \underline{t}_j$, if $i \neq j$ with $i, j \in \{1, \dots, M\}$.

It is possible that more sequences T that verify conditions 1.-3. exist (but also no sequence could exist, as explained afterwards) and therefore, those that minimize a specified cost function should be chosen.

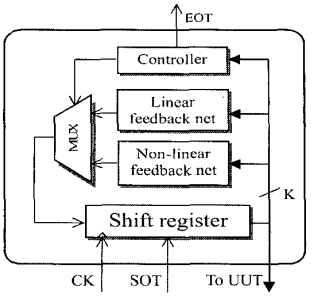

Fig.2 : The proposed Hybrid TPG architecture

3.2 Definitions

Let $\underline{x} \equiv (x_1 , x_2 , \dots , x_K)$ and $\underline{y} \equiv (y_1 , y_2 , \dots , y_K)$ be two *not completely specified* vectors of K elements, that is each vector element $\in \{0, 1, -\}$ where '-' indicates a don't care. Two elements x_i and y_i are equal ($x_i = y_i$) if and only if they assume the same value or at least one is a don't care.We define that $\underline{x}$ is *equal* to $\underline{y}$ ($\underline{x} = \underline{y}$) if and only if $x_i = y_i$ where $i \in \{1, \dots , K\}$.

We define that $y_i \subseteq x_i$ if and only if, when y_i is specified x_i, is not specified or is equal to y_i, or when y_i is not specified also x_i is not specified. The vector $\underline{y}$ is *included* in $\underline{x}$ ($\underline{y} \subseteq \underline{x}$) if and only if $y_i \subseteq x_i$ where $i \in \{1, \dots , K\}$. The *right minimum distance* between $\underline{x}$ and $\underline{y}$ is defined as:

$$d_{R,min}(\underline{x}, \underline{y}) = \min_h \{ h \geq 0 : \underline{x}[1, K\text{-}h] = \underline{y}[h+1, K] \}$$

where $\underline{x}[1, K\text{-}h] \equiv (x_1 , x_2 , \dots , x_{K-h})$.

It is simple to verify that $d_{R,min}(\underline{x}, \underline{y})$ is always $\leq K$. Let us assume that $d_{R,min}(\underline{x}, \underline{y})$ is d, and that $\underline{x}' \subseteq \underline{x}$ and $\underline{y}' \subseteq \underline{y}$ are two vectors obtained specifying some of the don't care of $\underline{x}$ and $\underline{y}$ respectively. It is easy to demonstrate that one and only one couple $(\underline{x}', \underline{y}')$ exists such that the following two conditions are verified:

1) it is possible to load a right shift register with x', and then, serially introducing d bits in the shift register, reach y' as the state of the shift register;
2) x' and y' have the maximum number of don't cares.

In the following $\cup$ is the ordered juxtaposition operation between sequences and the bits that have to be specified to obtain minimum distance are reported in bold. It is always possible to single out a unique sequence of d+1 vectors $(\underline{e}_0, \underline{e}_1 , \underline{e}_2, \dots, \underline{e}_d) = \underline{x}' \cup I \cup \underline{y}'$, where I is the sub-sequence of the d–1 intermediate vectors linking $\underline{x}'$ with $\underline{y}'$ and $d_{R,min}(\underline{e}_i, \underline{e}_{i+1}) = 1$ with $i = 0 \dots d$–1. As example, if $\underline{x} \equiv$ (-110-) and $\underline{y} \equiv$ (1--0-), $d_{R,min}$ is equal to 3. The couple $(\underline{x}', \underline{y}')$ is $\underline{x}' \equiv$ (**0**110-) and $\underline{y}' \equiv$ (1--**01**), and in this case $(\underline{e}_0, \underline{e}_1, \underline{e}_2, e_3)$ is ($\underline{x}' \equiv$(**0**110-), (-0110), (--011), $\underline{y}' \equiv$(1--**01**)).

3.3 Proposed synthesis algorithm

The set problem of Section 3.1, is a combinational problem that can be faced and effectively resolved using the method of the "state space search" [9]. The search process consists in generating and inspecting a graph, in which every node represents a partial solution of the problem, and the successors of a node represent possible steps toward a solution. The search of a solution corresponds to the search of an appropriate path between an initial node and a final node of the graph, that minimizes a predefined cost function.

The procedure to find out a solution sequence T of the set problem will be described referring to the RSR case. Note that the LSR procedure is equal to the RSR case, if each vector $\underline{x}_i$ is substituted with the reversed vector.

Every node P of the search graph:

- represents the partial sequence of T built up to that point, and this sub-sequence contains a subset of N-k vectors included in vectors of T_S and the intermediate linkage vectors;
- is characterized by $\underline{x}_{P,Last}$ that is the last vector added to its sub-sequence and it is a vector of T_S with some of the dont'cares specified if necessary;
- is characterized by the k vectors of T_S not used yet, $\{\underline{x}_{P,1}, \ldots, \underline{x}_{P,k}\}$.

The initial graph nodes are N and everyone corresponds to a sequence composed by only one vector of T_S. In general, starting from P, k successor nodes are generated adding one of the remaining k vectors of T_S $\{\underline{x}_{P,1}, \ldots, \underline{x}_{P,k}\}$. The number of different sub-sequences linking a node to a successor is not finite. Proposed algorithm uses the minimum length of the intermediate sub-sequence given by $d_{R,min}-1$.

To go from node P to one successor P', the minimum right distance between the last vector of P $\underline{x}_{P,Last}$ and $\underline{x}_{P,j}$, one of remaining vector of T_S, must be calculated. These two vectors are specified in order to minimize their distance (as reported in Section 3.2), and then they are added to the sub-sequence of P with the linking vectors, achieving the sub-sequence of P'. The last vector of P' $\underline{x}_{P',Last}$ is $\underline{x}_{P,j}$ appropriately specified.

A node of the search graph is eliminated if it corresponds to a partial sequence T in which a completely specified vector is present twice. Furthermore, if an intermediate vector can be specified to get a not yet used vector of T_S, it can be demonstrated that such path is not optimal. A final node of the search graph, associated with an ordered sequence $T = (\underline{t}_1, \underline{t}_2, \ldots, \underline{t}_M)$, in which all the vectors of T_S have been used, without repetitions of completely specified vectors, is a solution of the problem.

A simple example:

In the following a simple example, that expands a single starting node, is presented. Let consider the case in which the test set for hard faults is T_S = {$\underline{x}_1$≡(0--10), $\underline{x}_2$≡(-110-), $\underline{x}_3$≡(1--0-), $\underline{x}_4$≡(00-0-)}, composed by N = 4 vectors, K = 5 bit long. The search graph, starting from the vector $\underline{x}_1$≡(0--10), is shown in Fig. 3. Every node P in Fig. 3 contains $\underline{x}_{P,Last}$ that is the last vector of T_S (if necessary, appropriately specified) added to T, and is characterized by a pair of numbers in the form a/b where a is the T sequence length up to the node and b is the number (k) of remaining vectors of T_S. The distance value is d, reported beside the edge from P to P'. Vectors beside the edges are the last vector T_S belonging to sub-sequence of P, after the specification of some don't cares, and the d–1 intermediate vectors. Starting from $\underline{x}_1$≡(0--10), the minimum distance from the remaining three vectors (that are $\underline{x}_2$, $\underline{x}_3$, $\underline{x}_4$) and the intermediate vectors to go from $\underline{x}_1$ to $\underline{x}_2$, $\underline{x}_3$, $\underline{x}_4$ are calculated. In this way three new nodes of the graph are generated. A solution is a path on the search graph composed of N valid nodes. Inspecting the graph of Fig.3 six solutions exist, but only one of them minimizes the number of used vectors. Optimal sequence length is five and the relative path is highlighted in the figure.

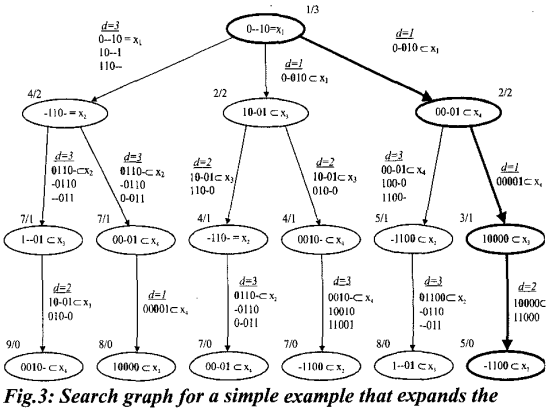

Fig.3: Search graph for a simple example that expands the single node $\underline{x}_1$≡(0--10) of T_S

In general, the number of possible solutions is limited by $N!$: it is not certain, in general, that a solution node exists but the big number of potential solutions suggests, and the following experiments confirm, that the case of absence of solutions is extremely infrequent.

The total graph is composed by a number of nodes limited by:

$$\sum_{i=1}^{N} N \cdot (N-1) \cdots (N-i+1)$$

It is impossible to inspect the search graph entirely and hence different heuristic techniques were exploited. One heuristic inspects nodes that have a better ratio between T sequence length and the number of vectors of T included in T_S (uniform cost function approach), another uses a deep-first approach following the path with minimum distance between each node and its successors. The latter showed to yield better performances and it is used in experimental results of Section 4.

However, several don't cares can remain in the final sequence T and they must be completed, avoiding repeated vectors. A procedure for the search of valid don't cares completion has been predisposed. Let suppose R the number of don't cares to specify, a possible solution could be to consider the 2^R sequences without don't cares and to verify which of these are correct. Proposed solution is the use of evolutionary algorithms [10]. In this paper the paradigm known as "selfish gene" was used. With regard to don't cares completion problem, for what concerning the implementation of a genetic algorithm, the optimization function associates to R-dimensional vectors (in which *i-th* bit states *i-th* don't care in the sequence), the number of repeated vectors in completely specified sequence obtained in this way [11].

Once a solution sequence $T = (\underline{t}_1, \ldots, \underline{t}_M)$, composed of vectors without don't cares, is found, it is necessary to synthesize the truth table of the corresponding feedback net. Supposing that the shift register is initialized to $\underline{t}_1$, for every vector $\underline{t}_i$, with i = 1…M-1, the row of the truth table corresponding to the vector $\underline{t}_i$ must have as output bit, the bit that is needed to achieve the following vector $\underline{t}_{i+1}$ shifting in the right direction. This bit is the most significant one in $\underline{t}_{i+1}$. Remaining rows of the truth table are not specified.

4. EXPERIMENTAL RESULTS

In order to evaluate Hybrid TPG performances, benchmark circuits and test vectors must be used. In this paper performance comparison of different architectural solutions for TPGs is made with reference to ISCAS'85 circuits [12]. Test vectors for the experiments reported in Table 1 are the same used in [13] and are available at http://www.die.unina.it/nlfsr/vectors.html.

Comparison of TPG performances is made with reference to test sequence length. Above described search algorithm has been implemented in C language and run with above mentioned test sets. Results refer to the best solution obtained using both right and left shift registers, CPU times refer to a Pentium II 350 Mhz. In Table 1 sequence length and required CPU time to devise T sequence able to detect the hard faults reported in [13] for ISCAS'85 circuits, is shown. Hybrid TPG sequence length is compared with sequence length for previously proposed TPGs [13-15].

Table 1 shows that Hybrid TPG provides reduced T sequence length with respect to previously proposed techniques. Main advantages of proposed architectures are:

1) the truth table of the combinational net used in Hybrid TPG is formed by few specification points. As a consequence the TPG is expected to require small silicon area and provide low power dissipation during off-line or on-line testing;
2) the short length of the testing sequence, corresponding to the number of clock cycle needed to complete the testing sequence, allows short testing time, and a low probability of aliasing in the signature analysis;
3) proposed technique uses one only initialization vector (usually referred as seed) and does not need extra memory to store the seeds.

Test set derived for ISCAS85			Other techniques			Proposed solution
	K	N	[15]	[14]	[13]	
c432	36	6	14.971	1,619	125	32 (0.12s)
c499	41	14	$1.464 \cdot 10^9$	15,862	22,064	167 (0.12s)
c1355	41	12	$6.473 \cdot 10^9$	$29.8 \cdot 10^6$	$122.8 \cdot 10^6$	394 (0.17s)
c1908	33	14	4.520	2,181	1,169	331 (0.36s)
c3540	50	22	$135 \cdot 10^6$	475,961	970	200 (0.36s)
c6288	32	36	$2.804 \cdot 10^9$	$4.39 \cdot 10^6$	$98 \cdot 10^6$	506 (0.95s)

Table 1 Results with pseudo-random testing completed by deterministic vectors for hard faults

5. CONCLUSIONS

In the paper a novel test pattern generator for Built-In Self-Test technique of testable combinational circuits based on a reconfigurable circuit is proposed. A synthesis tool that exploits heuristic search and the "selfish gene" genetic algorithm to determine test pattern generator for a given test set, is also proposed. Experiments are conducted on well-known ISCAS'85 circuits. Architecture performances are evaluated with respect to synthesis time and test sequence length. The experiments showed that proposed architectures yield reduced test vectors sequence length when compared with other architectures proposed in the literature.

6. ACKNOWLEDGES

Authors would thank professors Dimitrios Kagaris, Spyros Tragoudas and Amitava Majumdar for kindly providing us test vectors needed to detect hard faults and to compare proposed TPG with previously proposed ones.

REFERENCES

[1] M. Abramovici et al.: "Digital systems testing...", Computer Science Press, New York, NY (USA), 1990.
[2] P. H. Bardell, W. H. McAnney et al.: "Built-in Test for VLSI: Pseudo-Random Techniques", New York: Wiley, 1987.
[3] B. Vasudevan et al.: "LFSR Based Deterministic... ", Digest of papers of VLSI Test Symposium, 1993, pp.201-207.
[4] C. Dufaza, G. Cambon: "LFSR based Deterministic and Pseudo-random...", 2nd ETC, Munich, pp.27-34, 1991.
[5] D. K. Pradhan, M. Chatterjee: "GLFSR—A New Test Pattern Generator for Built-in-Self-Test", IEEE Trans. on CAD, vol. 18, no. 2, Feb. 1999, p.238.
[6] W. Daehn and J. Mucha: "A hardware approach to...", IEEE Trans. on Comp., vol. C-30, no. 11, pp. 829-333, 1981.
[7] L.T.Wang, E.J.McCluskey: "A Hybrid design of Maximum...", Int'l Test Conference, Sept. 1986, pp 38-47
[8] S.W. Colomb: "Shift register Sequences, revised edition", Aegean Park Press, Laguna Hills, CA, 1982.
[9] N.J. Nilsson, 'Problem-Solving Methods in Artificial Intelligence", McGraw-Hill, N.Y. USA, 1986.
[10] D.E. Goldberg: "Genetic Algorithms in Search, Optimization, and Machine Learning", Addison-Wesley, 1989.
[11] F.Corno et al.: "The Selfish Gene... ", SAC'98: 13th Annual ACM Symposium on Applied Computing, Atlanta, Georgia (USA), February 1998, pp. 349-355.
[12] F. Brglez and H. Fujiwara, "A neutral netlist...", Proc. IEEE Int. Symp. Circuits and Systems, pp 663-698, 1985.
[13] D. Kagaris et al.: "On the Use of Counters...", IEEE Trans. on Computers, vol. 45, no. 12, pp. 1405-1419.
[14] A.Majumdar: "WRAPTure: A Tool for Evaluating...", Proc. IEEE Int. Conf. Computer Design, pp. 288-291, 1994.
[15] F. Muradali et al.:"A New Procedure for Weighted...", Proc. Int. Test Conference, pp 1013-1022, 1993.