Efficient Random Testing with Global Weights

Arno Kunzmann
Forschungszentrum Informatik (FZI)
Haid-und-Neustrasse 10-14
D-76131 Karlsruhe, Germany
Phone +49 721 9654456
Telefax +49 721 9654409
Email kunzmann@fzi.de

Abstract

This paper describes a new and highly efficient approach for weighted random pattern generation. In contrast to the state-of-the-art approaches, where input specific weights are computed, the proposed method is based on the computation of global weights. This set of a very few weights (e.g., 4 or 8) is pattern oriented and therefore, with each weight the generation of the related random patterns is uniquely specified. Starting with a deterministic test pattern set and the inherent pattern specific weights, columns or rows can be inverted such that the initial weights are maximized in order to minimize the number of random patterns. Our experiments with the prototype system POWER-TEST (Pattern Oriented WEighted Random TESTing) show that very high fault coverage can be achieved with low computation and implementation effort at low self-test hardware costs.

1 Introduction

With decreasing transistor dimensions and upcoming significant wire delays, the importance of self-test techniques is increasing, since a dynamic test application at system speed is essential for the identification of dynamic faults [1]. Here, the self-test of digital circuits with randomly generated patterns is one of the most efficient test strategies in a production environment [2, 3]. Especially, the extensions towards weighted random pattern (WRP) testing significantly increases the test efficiency by reducing the number of random patterns, which is necessary to achieve a target fault coverage. All these methods require both a technique for calculating the specific input weights and an appropriate self-test hardware for the generation of the weighted random patterns. Each weight represents the probability that the corresponding primary input is set to the logic value 1.

There exists a relatively large number of proposals [e.g., 2, 4, 5, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16] how an optimized set of weights can be determined such that the required number of random patterns is minimized. In all these approaches the input-oriented weight computation can be regarded as a common feature and strategy. In contrast to these input-oriented approaches, the proposed new method aims at calculating global, pattern-oriented weights. Consequently, all inputs are stimulated with the same, currently selected weight or, in the case of an inverted input, with its complementary value.

In this paper we will show that the proposed method of calculating and applying global weights for random testing is very efficient, i.e., at low self-test hardware costs, computation time and implementation effort. A high fault coverage can be reached with significantly reduced pattern counts.

In the following section the state-of-the-art in WRP testing will be discussed by giving a brief classification and summary of the different weight calculation approaches. Then, based on a small example the basic ideas of the new approach will be introduced. A detailed description of the weight calculation algorithm can be found in section 3. The proposed WRP test generation system has been implemented as part of the test system INSPIRATION (Incomplete Scan Path Integration). In section 4, experimental data underline the efficiency of the proposed method.

2 Motivation

One of the major benefits of WRP testing is, compared to a test with stored (e.g., deterministic) patterns, the drastically reduced data volume, since basically only the weights for each input have to be stored. In addition, the number of random patterns has been significantly reduced with the introduction of unequiprobable weights. Additional pattern count reductions can be obtained with several weight distributions instead of one single. Several papers address this extension in recent publications.

WRP generation methods and techniques can be classified into functional and topological based [e.g., 2, 4, 5, 6] and deterministic test pattern based strategies [e.g., 7, 8, 9, 10, 11, 12, 13, 15, 16]. In spite of some hypotheses in [16] that the latter strategy is more efficient with respect to the required test length, both strategies have shown to

EURO-DAC ’96 with EURO-VHDL ’96
0-89791-848-7/96 $4.00 ©1996 IEEE

significantly reduce the random pattern count. The commonly applied technique of using several weight distributions has the inherent drawback of an increased self-test hardware overhead.

In contrast to the discussed state-of-the-art approaches, the new method aims at the computation of one single and global weight that can be used for all inputs. Starting point is a deterministic test pattern set that may contain also unspecified bit positions. The initial global weight for stimulating the inputs with a logical "1" is the relation between the number of logical "1" and the number of bit positions that are unequal to the don't care value.

In order to increase the probability for covering a given test pattern, two manipulations will be introduced: (1) row inversion and (2) column inversion. Both inversion techniques can be applied to increase the number of 1's. By the proposed iterative inversion technique (cf. section 3) the number of 1's and the fault detection probability can be maximized. With each row inversion, the percentage of completely inverted random pattern increases proportionally. With any column inversion, the corresponding input signal has to be inverted.

Figure 1 shows an example for an initial deterministic pattern set (a) and the resulting pattern set (d). In more detail, the interim result (b) is obtained from (a) by inverting rows 2 and 5, an inversion of the second line in (b) yields (c), and finally rows 3 and 4 will be inverted to get the result (d). Thus, the weight could be increased from initially 11/18 to 15/18. As validated by several fault simulation phases instead of 221 random patterns in average only 65 patterns are sufficient.

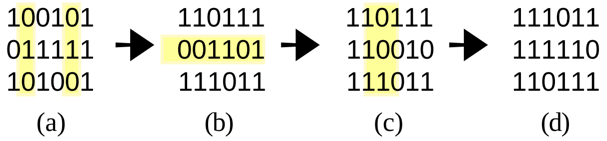
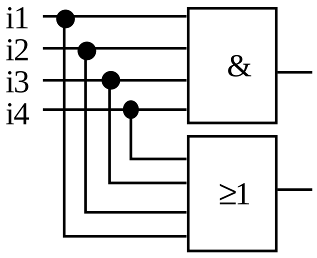


Figure 1: Example for row and column inversions

Figure 2 shows an often cited situation where conventional optimization algorithms fail in producing efficient results.



i1	i2	i3	i4
1	1	1	1
0	1	1	1
1	0	1	1
1	1	0	1
1	1	1	0
0	0	0	0
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

Figure 2: AND-/OR-gate conflict and related pattern set

In this situation, no columns will be inverted since the overall number of 1's can not be increased by this operation. But obviously, an inversion of rows 6 to 10 would significantly increase the global weight from initially 0.5 to 0.8 (=32/40). Consequently, the necessary number of random patterns will be decreased, assuming an appropriate WRP generation technique.

3 Weight Determination

Based on a set of partly specified deterministic test patterns, where some bit positions have not been set to either 1 or 0, the main goal of the weight computation is to improve the efficiency of random test pattern generation. Given the confidence C representing the probability that all faults are detected by N random patterns, and the fault detection probability p_f, the number of required random test patterns N can be calculated by

$$(1) \qquad N = \ln (1-C) / \ln (1-p_f).$$

Since C is fixed, for instance at the value 0.999, only the parameter p_f can be modified. Given a deterministic test pattern d ($d_1,...,d_n$) of width n and a global probability P for generating random patterns, p_f can be expressed by

$$(2) \; p_f(d)= \prod DP_i \begin{cases} DP_i:=P & \text{if } d_i=\text{"1"} \\ DP_i:=(1-P) & \text{if } d_i=\text{"0"} \end{cases} \quad i=1,...,n$$

In Figure 3, the effect of increasing the global probability P is illustrated by a small example. Here, it is assumed that the pattern width n is 10, all values are specified and at the x-axis, the global probabilities are given, ranging from 0 to 1. The two extreme values are related to a test pattern with only 0's and 1's, respectively. The other eight values assume that a test pattern is given with the appropriate number of 1's and 0's. The y-axis shows the resulting test length N to cover such a test pattern, assuming the confidence C=99.9%. As indicated in Figure 3, with increasing distance to the 0.5 value, the number of necessary random patterns decreases.

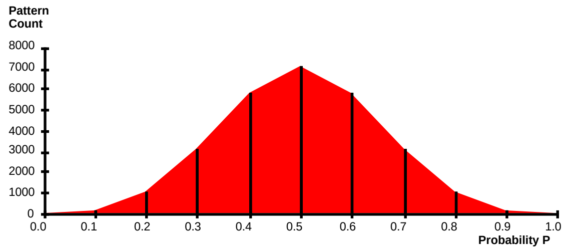


Figure 3: Effect of varying the global probability P

3.2 Maximizing the Global Input Probability

In order to use this effect to increase of the random test efficiency, a given deterministic test set has to be modified with the major objective to increase the number of 1's and the global input probability P.[*] To modify the pattern set two different inversion techniques and phases can be distinguished:

(Ph1) column inversion
(Ph2) row inversion.

Both strategies, applied on a basic test pattern set, have direct relations with the corresponding self-test hardware: While strategy (P1) implies that the related input signals of the circuit-under-test (CUT) will also be inverted during the test application phase, row inversion (2) means that random test patterns have to be completely inverted before its application.

ad (Ph1): Starting point of the column inversion procedure is a given deterministic test pattern set. Then, for any bit position it is checked whether its inversion will yield an increased number of 1's. Whenever this is the case, the related input of the CUT has to be inverted for the test application or, if already marked as inverted, its inversion will be omitted. Obviously, the order of analyzing the input positions is arbitrary, assuming a complete analysis of all inputs.

ad (Ph2): Assuming an initial column inversion, the modified test pattern set has to be analyzed. Any test pattern is assumed to be inverted, if its 0-count is greater than its 1-count. Similar to phase (1), already marked patterns can be omitted.

The introduced inversion phases are related to each other, i.e., column inversion(s) may allow new row inversions and vice versa. Therefore, the inversion procedure is only stopped if two consecutive inversion phases do not enhance the global value P. The basic procedures are given in Figure 4.

```
procedure row_inversion;
begin
for i:=1 to the_number_of_patterns do
  if count1h[i]<count0h[i] then          (* horizontal count *)
      begin invert(pattern[i]); pattern_set_is_changed:=true;  end
end;
procedure line_inversion;
begin
for i:=1 to pattern_width do
  if count1v[i]<count0v[i] then          (* vertical count *)
      begin invert(pattern[i]); pattern_set_is_changed:=true; end
end;
procedure maximize_global_weight_P;
begin
  repeat            pattern_set_is_changed:=false;
                    row_inversion;
                    line_inversion
  until pattern_set_is_changed=false;
end;
```

Figure 4: The Basic inversion procedures

Since the maximization of P is only an indirect measure of the required number of random test patterns, also different optimization strategies have been analyzed. For instance, according to equation (1) with each inversion the current number of theoretically required random patterns can be re-calculated. According to our evaluation results, this strategy did not yield the expected efficiency improvements, mainly due to the following reasons:

(a) The given deterministic pattern set is partly unspecified and gives some indications which patterns are hard to detect (e.g., if only a few bit positions are unspecified) or can be easily identified (e.g., if only a few bit positions are set to 1 or 0). Unfortunately, this is only an sign since in general several alternative test patterns exist to cover the same fault, and therefore, the number of specified bit positions can significantly vary.

(b) The optimization strategy is dominated by a very few test patterns with a low number of unspecified bit positions, simultaneously neglecting the other patterns. A re-calculation of the pattern count would only be reasonable, if further analyses of these dominating test patterns are performed in order to validate the uniqueness of the patterns.

Due to these reasons, the weight determination has been performed without any re-calculation of the random pattern count N, but in order to prefer the treatment of deterministic test patterns with a large number of specified values, a threshold L has been introduced: Deterministic test patterns are evaluated in phase (Ph1) and (Ph2), only if its number of specified bit positions exceeds this value L.

3.3 Weight Set Computation

Based on the modified deterministic test pattern set and on the related scan-path inversions, one could use the resulting global input probability for the random pattern generation. But since this leads to only a slight reduction of the pattern count in general, the following discussion is focused on the computation of weight sets.

Assuming the possibility to generate random patterns with weights in the interval [0,1], the optimum weight of each single pattern is given by the quotient between the number of 1's and its number of specified bit positions. Conversely, assuming the granularity of 1/8 and 9 basic weights, respectively, the number of patterns requiring these weights is also known. This information can be used to identify a set of a very few weights to improve the efficiency of the produced random patterns. The discussion of this effect is also part of the enclosed experimental results (cf. section 5), the influence on the self-test hardware structure will be discussed in the subsequent section.

[*] This strategy is assumed for the result of this paper without loss of generality, since the maximization of the number of 0's and the minimization of P is a dual problem leading to the same efficiency increase.

Input of the optimization procedure is a user-defined maximum number of weights, typically 2, 4 or 8, which will be iteratively selected during the test generation phase in order to improve the test quality and to ensure the realization of a cost-efficient self-test hardware. It should be noted that single weights can be applied more than once per iteration cycle if a lot of test patterns would require this weight. The basic procedure for calculating these weights is given by Figure 5.

```
...
read(the_number_of_selectable_weights);
...
procedure get_weight_set;
(*  Input is an array "count_ws" of length
"the_number_of_basic_weights", where for each weight the
number of related deterministic test patterns is stored            *)
begin
divisor:=number_of_testpatterns/number_of_selectable_weights;
for i:=1 to the_number_of_basic_weights
        weight_count_for_ws[i]:=round(count_ws[i]/divisor);
end;
```

Figure 5: Basic procedure for the weight set computation

After having read the input data, a divisor is computed which is determined by the quotient between the test pattern count and the number of selectable weights. The variable *count_ws[i]* contains the number of test patterns requiring the i-th basic weight. Divided by the divisor, the variable *weight_count_for_ws[i]* indicates, how often the basic weight i will be applied within the pattern generation phase. Obviously, a weighting of the weights is performed by this number.

In order to illustrate the weight set computation and optimization procedure, the variables *the_number_of_selectable_weights* and *number_of_testpatterns* are set to 8 and 136, respectively. Consequently, *divisor* is 17. Assuming that *count_ws*[0,1,2,3,4,5,6,7,8] is set to [0,0,0,0,0,24,48,64,0], the resulting array *weight_count_for_ws*[0,1,2,3,4,5,6,7,8] will be [0,0,0,0,0,1,3,4,0]. In this case, half of the random test patterns will be generated with the input probability 7/8 and the other half is generated according to the relation 1:3 with input probability 5/8 and 6/8.

4 Self-Test Hardware

According to the described weighting technique, the required self-test hardware can be easily realized by a single LFSR and a conventional weight calculation circuit, plus the iterative selection of the necessary weights. The basic structure of the proposed self-test hardware is given in Figure 6, where one 4:1 multiplexer is used to toggle between the three probabilities 0.625, 0.75 and 0.875. By using the value 0.875 twice, half of the generated random test patterns are generated with this weight that is obviously very important to achieve a high fault coverage.

The hardware overhead imposed by the proposed self-test structure is relatively low, since only a very few gates are needed for the weight computation, one multiplexer is required and, depending on its data width, one modulo-counter for iteratively selecting the probabilities has to be added. In contrast to the state-of-the-art approaches, where due to the input specific weights also correlating inputs have to be explicitly suppressed by introducing dedicated test hardware structures (e.g., [5]), in the proposed approach such correlations are excluded by construction.

According to the experimental results (cf. section 5) an 8:1-multiplexer and a 3-bit counter are sufficient to achieve an almost complete fault coverage. As also shown by the experimental data, the relative number of lines to be inverted is such low that an inversion of the multiplexer output is in almost all cases not relevant. Nevertheless, the realization of this inversion would only require one additional 'EXOR'-gate between the multiplexer and the scan-in input. Here, one input is the multiplexer output, the other input controls the inversion, e.g., by testing the value 0 of an appropriate modulo-x counter.

5 Experimental Results

The described algorithms have been implemented in the prototype system POWER-TEST (Pattern Oriented WEigthed Random TESTing) as submodules of the test system INSPIRATION (Incomplete Scan Path

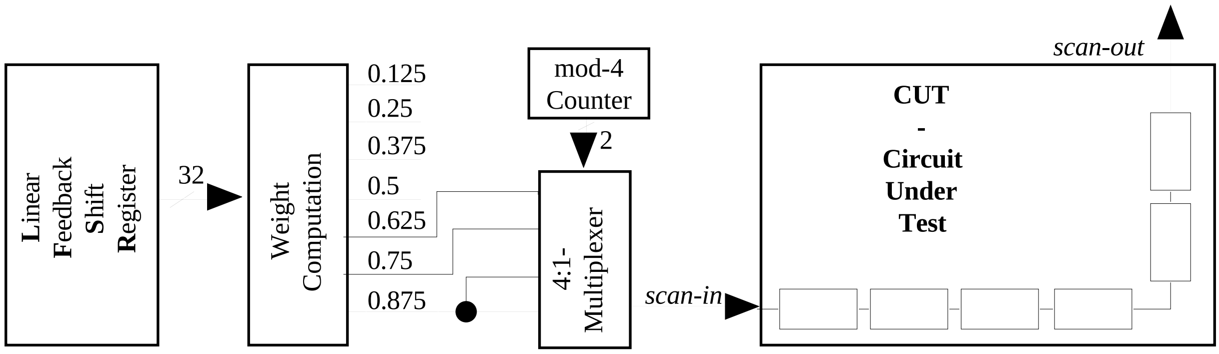


Figure 6: Basic structure of the self-test hardware

Benchmark Circuit	**Fault Count (detectable)**	**Random Pattern Count**	**Equiprobable Random Patterns**		**PROTEST: Input Oriented Weighted Patterns**		**POWER-TEST: Pattern Oriented Weighted Patterns**	
			FC [%]	***Undet. Faults***	***FC [%]***	***Undet. Faults***	***FC [%]***	***Undet. Faults***
c2670	2,408	2,000	88.0	290	97.9	50	97.6	51
		10,000	88.0	290	99.2	19 (31*****)	99.5	6
		50,000	88.4	279	99.8	6	99.9	2
c5315	4,895	2,000	99.9	6		n.a.	99.8	10
		10,000	99.9	3		n.a.	100.0	0
		50,000	99.9	3		n.a.		
c7552	6,876	2,000	94.7	365	97.6	165	96.8	212
		10,000	95.8	292	99.3	47	98.6	93
		50,000	96.8	218	99.6	25	99.5	29
s38584	30,237	2,000	94.8	1619		n.a.	96.8	844
		10,000	98.1	727		n.a.	99.3	178
		50,000	99.1			34******	99.9	31

***** [6] after 7,554 weighted random patterns (input oriented); ****** [6] after 42,426 weighted random patterns (input oriented)

Table 1: Comparison of fault coverage with different random pattern sets

IntegRATION) developed at the University of Karlsruhe and FZI [17]. Based on the well-known ISCAS benchmark circuits, the necessary basic set of partly specified deterministic test patterns has been computed with the integrated ATPG tool SPROUT-9v. Since the current prototype implementation is tuned on combinational circuits, it is assumed that sequential circuits offer a complete scan design.

In order to show the efficiency of the new approach, we have initially selected all benchmark circuits that are hard to be tested with equiprobable random patterns, i.e., after 50,000 random patters with input weight 0.5 some undetectable faults are left. For each circuit several analyses have been performed with different initial values (e.g., for the basic LFSR), and the below given numbers are the median values of these runs. The random pattern sequences have been produced by a LFSR with characteristic polynomial $x^{31}+x^{3}+1$. In Table 1, all evaluation results are based on 8 weights, i.e., the self-test hardware requires an 8:1-multiplexer for weight distribution. The introduced threshold value L has been set to 12, i.e., at least 12 bit positions have to be specified, otherwise the related test pattern will not be taken into account. In addition, the line inversion has been turned off to discuss its effect separately at the end of this section.

Table 1 provides a detailed look on some benchmark circuits by showing the achieved fault coverage during the fault simulation phase, and offering a direct comparison to the input oriented weight optimization by PROTEST [4]. As indicated by the achieved fault coverage and the remaining undetected faults, the savings are significant in the test pattern count compared with equiprobable patterns, and the achieved test quality is in the same range as provided by PROTEST. It should be noted that a main advantage of the POWERT-TEST system is a simplified weight computation and also a reduced test hardware effort compared to input oriented test methods.

Table 2 provides the evaluation results for additional sequential benchmark circuits where conventional random testing would require the application of more than 10,000 patterns. Here, the remaining undetected faults are given after having applied 10,000 equiprobable and weighted test patterns, respectively. In order to show the effect of varying the weight count, the optimization results are given for four and eight weights. In the latter case, for the benchmark circuit s526 (s641, s713) complete fault coverage could be achieved already after 4255 (8854, 8854) weighted patterns. The achieved results underline the efficiency of the proposed WRP strategy, and also show that an increased number of weighs does not guarantee and improved fault coverage and a reduced pattern count.

Finally, it should be noted that all above given experimental data are purely based on row inversions, since the percentage of required line inversions typically was lower than 5%. Therefore, for almost all benchmark circuits the fault coverage did not show measurable improvements when incorporating line inversions. Consequently, an inclusion of the required extra logic in the self-test hardware for a periodical inversion of complete test patterns need not be realized to obtain the shown improvements.

Circuit	Fault Count (detectable)	Undetected Faults after 10,000		
		Equiprobable Random Patterns	Weighted Patterns (POWER-TEST) with	
			4 Weights	8 Weights
s420	356	37	1	1
s526	491	2	0 (2130)	0 (4255)
s641	406	11	0 (5043)	0 (8854)
s713	470	11	0 (6377)	0 (8854)
s820	773	9	4	4
s832	775	17	4	2
s838	711	127	30	31
s953	893	10	1	1
s1196	1,044	25	18	17
s1238	1,051	16	13	12
s1423	1,241	7	1	2
s5378	3,837	51	28	22
s9234	5,564	801	193	190
s13207	8,386	688	196	204
s38584	30,237	727	211	178

Table 2: Experimental results without line inversions

According to the experimental data, an inversion rate of more than 20% is required before significant coverage improvements can be achieved with line inversions. This is the case for circuits s420 and s838 with a rate of 22.4% and 21.5%, respectively. As indicated in Table 3, the number of random patterns and aborted faults could be significantly reduced: for circuit s420, the pattern count could be decreased below the 10,000 limit. For circuit s838, the number of undetected faults could be halved.

Circuit	Line Inversion Rate	Undetected Faults after 10,000		
		Equiprobable Random Patterns	Weighted Patterns (POWER-TEST) with	
			4 Weights	8 Weights
s420	22.4 %	37	0 (3651)	0 (6122)
s838	21.5 %	127	12	15

Table 3: Improved results including line inversions

6 Conclusions

In this paper, a new and very promising approach for an efficient weighted random pattern test has been introduced. In contrast to the state-of-the-art approaches, where for each circuit input specific weights are determined, the proposed method is pattern oriented. The evaluation of the prototype system POWERT-TEST shows significant pattern count reductions over the equiprobable random pattern test. Direct comparisons with conventional input-oriented methods show a very similar optimization quality. By construction, both the basic weight generation procedures and the related self-test hardware can be realized at very low implementation effort.

References

[1] W. Maly: Future of Testing: Reintegration of Design, Testing and Manufacturing; Proceedings European Design and Test Conference, March 1996.

[2] P. H. Bardell, W. H. McAnney, and J. Savir: Built-In Test for VLSI: Pseudorandom Techniques; Wiley Interscience, 1987.

[3] R. W. Bassett et al.: Low Cost Testing of High Density Logic Components; IEEE Design and Test of Computers, 7(2):15-28, April 1990.

[4] H.-J. Wunderlich: Self-Test Using Unequiprobable Random Patterns; Proceedings Fault-Tolerant Computing Symp., pages 258-263, 1987.

[5] H.-J. Wunderlich: Multiple Distributions for Biased Random Test Patterns; Proceedings Int. Test Conference, pages 236-244, 1988.

[6] M. A. Miranda, C. A. López-Barrio: Generation of Optimized Single Distributions of Weights for Random Built-In Self-Test; Proceedings International Test Conference, pp. 1023 - 1030, 1993.

[7] E. B. Eichelberger et al.: Weighted Random Pattern Testing Apparatus and Method, U.S. Paten No. 4,801,870, January 1989.

[8] J. A. Waicukauski and E. Lindbloom: Fault Detection Effectiveness of Weighted Random Patterns; Proceedings Int. Test Conference, pages 245-255, 1988.

[9] J.A. Waicukauski et al.: A Method for Generating Weighted Random Test Patterns; IBM J. Res. Dev., 33(2):149-161, March 1989.

[10] J. A. Waicukauski and F. Motika: Testing VLSI Chips with Weighted Random Patterns; Proceedings Int. Symp. VLSI Tech., Systems and Applications, pages 149 -154, Taipei, Taiwan, May 1989.

[11] D. M. Wu and J. Waicukauski: Built-In Test Using Perturbed Deterministic Patterns; Proceedings Int. Symp. Circuits and Systems, Volume 3, pages 2232 - 2235, New Orleans, LA, May 1990.

[12] S. Pateras and J. Rajski: Cube-Contained Random Patterns and their Application to the Complete Testing of Synthesized Multi-Level Circuits; Proceedings Int. Test Conference, pages 473 -482, 1991.

[13] M. Bershteyn: "Calculation of Multiple Sets of Weights for Weighted Random Testing"; Proceedings International Test Conference, pp. 1031 - 1040, 1993.

[14] M. A. Miranda, C. A. López-Barrio: Generation of Optimized Single Distributions of Weights for Random Built-In Self-Test; International Test Conference, pp. 1023 - 1030, 1993.

[15] 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. Computer-Aided Design, 12(7):1050-1058, July 1993.

[16] R. Kapur et al.: Design of an Efficient Weighted Random Pattern Generation System; Proceedings International Test Conference, pages 491 - 500, 1994.

[17] A. Kunzmann and H.-J. Wunderlich: An Analytical Approach to the Partial Scan Problem; in "JETTA: Journal of Electronic Testing", Theory and Applications, 1, 1990.