Deterministic Pattern Generation for Weighted Random Pattern Testing

Birgit Reeb

University of Siegen

Hans-Joachim Wunderlich

CRC, Stanford University*

Abstract

Weighted random pattern testing is now widely accepted as a very economic way for external testing as well as for implementing a built-in self-test (BIST) scheme. The weights may be computed either by structural analysis or by extracting the required information from a precomputed deterministic test set. In this paper, we present a method for generating deterministic test patterns which can easily be transformed into weight sets. These test patterns contain only minimal redundant information such that the weight generation process is not biased, and the patterns are grouped such that the conflicts within a group are minimized. The quality of the weight sets obtained this way is superior to the approaches published so far with respect to a small number of weights and weighted patterns, and a complete fault coverage for all the ISCAS-85 and ISCAS-89 benchmark circuits.

1. Introduction

Various BIST architectures based on pseudo-exhaustive, random, weighted random, and deterministic patterns have been developed in recent years. They offer different tradeoffs between the test length and the hardware overhead required to achieve complete or sufficiently high fault coverage [1, 2, 4, 8, 9, 10, 11, 14, 25, 27]. In this paper, we address weight generation for random pattern testing which can be used either for a built-in test scheme [8,20,27] or for a built-off test scheme [23, 24].

Weighted random patterns may be applied to circuits where uniform pseudo-random testing would lead to an insufficient fault coverage. Such a case is shown in figure 1.

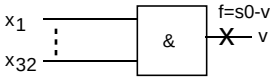


Figure 1: 32-input AND

The output v of the 32-input AND is tested for the stuck-at 0 fault (s0-v) if all the inputs x_i are set to 1. The occurence of such a pattern has a probability of 2^{-32} if all the inputs x_i are set to 1 with a probability of $p_i = 0.5$. Hence, the detection probability of the fault s0-v is $p_{so\text{-}v} = 2^{-32}$. If a fault f has a detection probability of p_f then $(1-p_f)^N$ is the probability that f is not detected by N independent patterns, and $C := 1-(1-p_f)^N$ is the confidence of detecting f by a random test set of N patterns. Finally, to reach a confidence C of a random test we need at least

$$N = \frac{\ln(1-C)}{\ln(1-p_f)} \qquad (1)$$

test patterns for the single fault f. Elementary transformations of formula (1) show that N grows linearly with $1/p_f$. Due to this observation, the test length required for detecting a set of faults F only depends on the faults $f \in F$ with lowest detectability p_f [21, 26]. For more details see [29].

Applying formula (1) to the s0-v fault f of figure 1, we need at least $4{,}48 \cdot 10^{10}$ patterns to obtain a confidence of 0.999. But if we set each primary input x_i to 1 with probability $p_i := \sqrt[32]{0.5}$ then the fault detection probability is $p_f = 0.5$, and $N = 600$ patterns will be sufficient.

We call $P := (p_1,...,p_n)$, $0 \le p_i \le 1$, a set of weights, and $p_f(P)$ is the detection probability of fault f if each input x_i is set to 1 with probability p_i. For the example of figure 1 with $P_1 := (\sqrt[32]{0.5}, ..., \sqrt[32]{0.5})$, we have $p_f(P_1) = 0.5$, and P_1 is an optimal set of weights. But unfortunately, there are circuits for which a single optimal set of weights does not work, as different faults may have contradictory requirements on the input probabilities.

An example is the circuit of figure 2 where the fault s0-v requires the weight set P_1 described above, but the fault

*On sabbatical leave from the University of Siegen,

Part of this work has been supported by the DFG under grant Wu 245/1-1.

ED&TC '96
0-89791-821/96 $5.00 ©1996 IEEE

s1-*w* requires the complementary value $P_2:=(1-\sqrt[32]{0.5}, ..., 1-\sqrt[32]{0.5})$, and there is no common weight set appropriate for both faults. The only solution is applying multiple sets of weights: first we generate $N_1=600$ random patterns corresponding to weight set P_1, then we apply $N_2=600$ random patterns corresponding to weight set P_2.

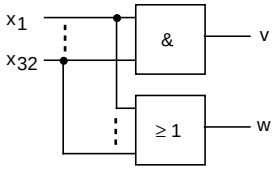


Figure 2: AND-OR requires multiple weights

In general, a circuit may require $P_1,...,P_r$ different sets of weights, and for each weight set P_i at least N_i random patterns have to be applied. The efficiency of multiple weight sets is determined by both the number r of different weight sets and the total number $N:=\sum_{i=1}^{r} N_i$ of random test patterns to be applied. For implementing a BIST scheme, the number r has most impact on the additional hardware costs, and N determines the test time.

The goal of weight generation is to reduce both r and N. In recent years, different methods for weight generation have been developed which can be classified roughly into methods based on structural analysis and methods based on deterministic test sets.

Methods based on structural analysis [5, 15, 16, 24, 26, 28, e.g.] use testability measures or heuristics to determine the weights. The advantage of these methods is the generation of very efficient weight sets in terms of the number of weight sets and the number of patterns. The drawback is that they may provide only an incomplete fault coverage as they are based on heuristics.

Methods based on deterministic test sets [13, 17, 18, 20, e.g.] use a precomputed test set $T: = \{t_1,...,t_M\}$ of M deterministic patterns. For each bit position i, the number of 1's is counted and divided by M providing the input weight

$$p_i := \frac{\sum_{j=1} t_j[i]}{M}. \qquad (2)$$

In a second step, random patterns corresponding to the weight set $P:=(p_1,...,p_n)$ are generated, and fault simulation is performed. If a certain amount of these random patterns does not detect any additional fault, random pattern generation and fault simulation are stopped. If the fault coverage obtained so far is not sufficient, a new deterministic test set is generated for the remaining faults, a new set of weights is computed, and the entire process is repeated.

The advantage of this procedure is that all non-redundant faults can be detected by a weighted random pattern test. The disadvantage consists in a high number of random patterns and a large number of weights. In the worst case each deterministic test pattern may result in a different weight set.

In [5], the trade-offs between the structural based method and the test set based method are discussed, and the most efficient weight sets known so far are published.

In this paper, we combine the advantages of both methods. It is shown that a complete fault coverage by a smaller number of random patterns and a smaller number of weights is obtained in comparison to the methods known so far if weight generation is already considered during computing the underlying deterministic test set.

The rest of this paper is organized as follows: in the next section we discuss the impact of deterministic test sets on weight generation. In section 3, an appropriate method for deterministic test generation is presented. The procedure that computes the weight sets is explained in section 4. As during BIST, the weights $p_i \in [0,1]$ have to be rounded the impact of this quantization is discussed in section 5. In section 6, experimental results on the ISCAS-85 and ISCAS-89 benchmark circuits are discussed [6, 7].

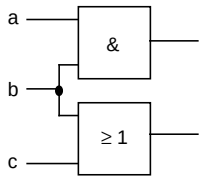
2. Weight generation and deterministic test sets

In general, a combinational circuit has many different complete deterministic test sets which result in different weight sets. Some deterministic test sets are less appropriate for weight generation due to mainly two reasons: the test sets may contain redundant information, or they may contain contradictory information.

Contradictory information is caused by a pair of test patterns with a large Hamming distance as for instance the two patterns (1,...,1) and (0,...,0) for the AND-OR of figure 2. They cannot be used both concurrently for weight generation. A method for resolving these conflicts is presented in section 4.

The problem of redundant information is explained with the help of the example in figure 3, the deterministic test sets T_I, T_{II} and the corresponding weight sets P_I and P_{II}.

The test sets T_I and T_{II} detect all 6 stuck-at faults at the primary inputs, but their weight sets lead to different fault detection probabilities as shown in table 1.



	T_I				P_I	T_{II}				P_{II}
a	1	0	1	0	$\frac{1}{2}$	1	0	1	1	$\frac{3}{4}$
b	1	1	0	0	$\frac{1}{2}$	1	1	0	0	$\frac{1}{2}$
c	1	0	0	1	$\frac{1}{2}$	0	0	0	1	$\frac{1}{4}$

Figure 3: Example circuit with test sets T_I,T_{II} and corresponding weight sets P_I and P_{II}

	s0-a	*s1-a*	*s0-b*	*s1-b*	*s0-c*	*s1-c*
$p_f(P_I)$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{3}{8}$	$\frac{3}{8}$	$\frac{1}{4}$	$\frac{1}{4}$
$p_f(P_{II})$	$\frac{3}{8}$	$\frac{1}{8}$	$\frac{15}{32}$	$\frac{15}{32}$	$\frac{1}{8}$	$\frac{3}{8}$

Table 1: Fault detection probabilities corresponding to the different weight sets P_I and P_{II}

As already mentioned, the minimum detection probabilitiy determines the test length. The minimum $p_f(P_{II})$ is $\frac{1}{8}$ whereas the minimum $p_f(P_I)$ is $\frac{1}{4}$ and P_{II} leads to a test length roughly twice as long as the length for P_I. The reason for this are some redundant bits set in T_{II}, which are transformed into "Don't Cares" by test set T_{III} (figure 4).

	T_{III}				P_{III}
a	1	0	1	X	$\frac{2}{3}$
b	1	1	0	0	$\frac{1}{2}$
c	X	0	0	1	$\frac{1}{3}$

Figure 4: Test set T_{III} and corresponding weight set P_{III}

If "Don't Cares" occur in a test set they must not contribute to the weights, and the input probability is computed as

$$p_i := \frac{\left|\left\{t \in T \mid t[i] = 1\right\}\right|}{\left|\left\{t \in T \mid t[i] \neq x\right\}\right|} \qquad (3)$$

In our example, this provides the weight set P_{III} shown in figure 4 and the fault detection probabilities shown in table 2. The minimum fault detection probability increases from $\frac{1}{8}$ for P_{II} up to $\frac{1}{6}$ for P_{III}.

In general, redundant bits can be removed from deterministic test sets in two ways. A test set containing a minimum number of patterns can be computed by a tool like COMPACTEST [19], or the number of specified bits can be reduced already during test generation. The experimental results of section 6 will show that deterministic test sets of minimum size may indeed provide more efficient weights, but in some cases they suffer from many conflicts between the patterns since nearly always all bits are specified.

	s0-a	*s1-a*	*s0-b*	*s1-b*	*s0-c*	*s1-c*
$p_f(P_{III})$	$\frac{1}{3}$	$\frac{1}{6}$	$\frac{7}{18}$	$\frac{7}{18}$	$\frac{1}{6}$	$\frac{1}{3}$

Table 2: Fault detection probability for the weight set P_{III}

Better results can often be obtained if ATPG tries to maximize the number of "Don't Care" bits of all test patterns as a large number of "Don't Care" bits reduces both redundancies in the test set and conflicts between test patterns. The next section presents a method for maximizing "Don't Cares" during ATPG.

3. Maximizing "Don't Cares" during test generation

As ATPG for combinational circuits is now well understood, only some modifications of a FAN-like algorithm [12] are sketched in the sequel. Both static and dynamic global implications and unique sensitization techniques are applied to accelerate the process of test pattern generation and the identification of redundancies [22]. Decisions are guided by a number of heuristics which particularly aim at generating test patterns with a large number of unspecified bits, and keeping the overall test set small. The heuristics use observability values which are determined by critical path tracing [3] and which are updated dynamically for each partial assignment.

For the propagation of fault effects, a node on the D-frontier is selected which is as close as possible to the primary outputs and is located on a path with a maximum number of undetected faults. To minimize the number of specified bits, the number of primary inputs which must be set to propagate the fault effect, is used as a second criterion.

For line justification, observability values are used if there is a choice of a gate input line to be set to a controlling value. Two cases are distinguished:

a) If the gate output is observable a gate input line is selected such that the number of undetected faults preceding this line is maximum.
b) If the gate output is not observable a gate input line is selected such that the number of primary inputs to be set becomes minimal.

In addition, the „bit flipping“ technique suggested in [19] is used to minimize the number of specified bits in a test pattern. For each specified bit the complementary logic value is simulated also. If the target fault is still detected, the bit position is considered as a " Don't Care".

4. Weight generation

For each fault, the previous algorithm produces an incompletely specified test pattern to be used for fault simulation and fault dropping as usual.

4.1 Integrating ATPG and weight generation

First, a certain number of pseudo-random patterns is generated and simulated. The detected faults are highly random pattern testable and can be dropped from the fault list *F*. For the remaining faults, redundancies are identified by a preliminary ATPG step. The redundant faults are removed from the fault list *F*, too, and for the rest of the faults weight generation is done by a cycle as shown in figure 5.

A) ATPG is performed for all the faults which are not detected by (weighted) random patterns so far. The ATPG tries to maximize the number of "Don't Cares" as described above. After the first run through the cycle there are already test patterns generated for all the faults, but these patterns are not used in successive runs since new test sets with less care bits may become possible.

B) The deterministic test set T of step A) is used for computing a set of weights using formula 2).

C) According to the weight set determined in B), random patterns are generated and fault simulation is performed. This process is controlled by a user-defined parameter *k*. Pattern generation is stopped if the last *k* successive patterns do not detect any new fault. A large value of *k* results in long test times and a small number of weight sets whereas a small *k* increases the number of weight sets and reduces the test length.

4.2 Resolving conflicts

The procedure presented so far may not terminate fast enough due to conflicts of test patterns. Conflicts are resolved by partitioning the test set. The Hamming distance $d(t_1,t_2)$ of two test patterns is the number of bits where these patterns are incompatible:

$$d(t_1,t_2) := \left\{ i \,\middle|\, \left(t_1[i] = 1 \wedge t_2[i] = 0\right) \vee \left(t_1[i] = 0 \wedge t_2[i] = 1\right) \right\}.$$

If $d(t_1,t_2) > m$ then there is no single set of weights such that both t_1 and t_2 are generated with a probability larger than 2^{-m}, and for large $d(t_1,t_2)$ it is not efficient to use both patterns simultaneously for weight computation.

We define $D(t,T,m) := \{ t' \in T \mid d(t',t) \le m \}$ and substitute the step B) of figure 5 as follows:

B') Weight computation by using Hamming distance.

A user defined parameter *m* controls the partitioning of the deterministic test set. It is reasonable to select $m = O(\log_2 k)$, and as already mentioned the pattern generation is stopped if the last *k* patterns do not detect a new fault. A test pattern $t \in T$ is determined such that $|D(t,T,m)|$ is maximum. Only the patterns from $D(t,T,m) \subset T$ are used for weight generation using formula (2).

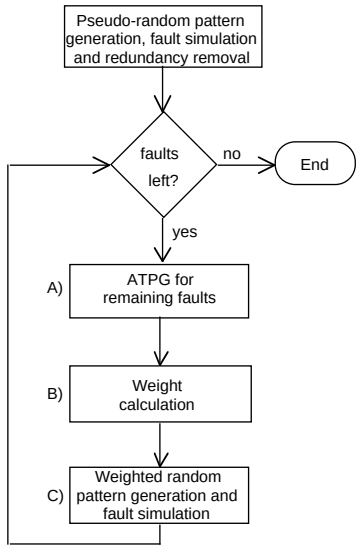


Figure 5: ATPG and weight generation

4.3 Reverse weight generation

Faults detected during the last run through the cycle of figure 5 are the hardest to detect faults, and the set of weights generated during this step corresponds to test patterns which have a high chance of detecting some additional faults also. Hence, all the faults are simulated by this last weight, and the cycles of the weight generation process are entered again for all the faults not detected by the last weight or by pseudo-random patterns.

Results of the reverse weight generation are only used if they provide an improved solution.

5. Impact of rounding the weights

The procedures B) or B') give us some real numbers for the weights $p_i \in [0,1]$, and there are no obvious methods for generating a random bit stream of these probabilities by hardware. For step C), a software approach as used in [5] can be applied: A 32-bit linear feedback shift register

(LFSR) with a primitive polynomial is simulated and cycles through 2^{32}-1 states, each state is interpreted as a dual fraction s=0.0...01 through s=0.1...11. The s are uniformly distributed in the interval $[0.0\ldots01_2, 0.1\ldots11_2]$. In each cycle one random bit for input x_i is generated; if $s \leq p_i$ holds then we set $x_i=1$ else $x_i=0$.

This technique provides nearly exact weights, but for an actual hardware implementation as a BIST scheme or a built-off test the weights have to be rounded [8, 13, 23, 24, 27]. The main idea of all these schemes is mapping r random bit streams to a weighted stream with the grid of 2^{-r}. For $r=3$, e.g., there exist boolean fuctions $f_1(a,b,c)$,..., $f_7(a,b,c)$ such that $f_i(a,b,c)$ becomes true for exactly $i \in \{1,2,\ldots,7\}$ minterms. Three independent input streams at a, b, and c provide an output probability of $\frac{i}{8}$ at $f_i(a,b,c)$. This scheme is sketched in figure 6, where the three independent input streams are generated by three different primitive LFSR's.

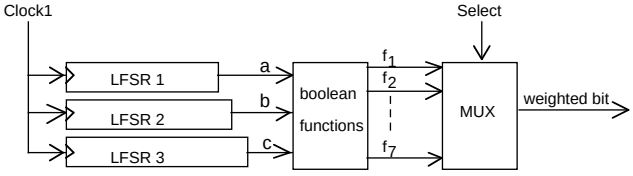


Figure 6: BIST scheme for weighted random test

In order to be applied in practice, the weighted pattern generation of step C) should consider a state of the art scheme as shown in figure 6 and has to use rounded weights. The experimental results of the next section will show that the proposed method works well even if a realistic rounding is used.

6. Experimental results

Experiments were performed on the set of combinational benchmark circuits and sequential benchmark circuits [6,7] which are assumed to be fully scannable. During the first experiments, exact weights are used. The weighted patterns were obtained as described above using a LFSR with the characteristic polynomial $x^{31}+x^3+1$. A new cycle of weight generation in figure 5 was started again if new faults were not detected after applying $k=1024$ consecutive weighted random patterns.

In the first experiment we investigated the effectiveness of a test set with a maximum number of "Don't Care" bits and a compact test set for weight generation. In these experiments, only the combinational benchmark circuits were used and a complete coverage of all non-redundant faults was required (table 3).

	maximum "don't care" test sets		compact test sets	
circuit	weights	test length	weights	test length
c432	1	1 512	1	1 490
c499	1	1 251	1	885
c880	1	796	2	10 552
c1355	2	4 397	1	6 588
c1908	3	5 452	3	12 557
c2670	3	9 289	2	9 600
c3540	3	3 575	5	22 718
c5315	1	2 138	3	15 170
c6288	1	74	1	1 012
c7552	6	17 046	21	38 216

Table 3: Number of weights and test lenghts by using different deterministic test sets ($k=1024$)

The results obtained by using maximum "Don't Care" bits are always comparable with the results for compact test sets, and in some cases they are distinctly better. Hence, in the following we use test sets with maximum "Don't Cares", and we call this method X-test. In the next experiment, the weight sets were computed without performing a prior pseudo-random test. Table 4a and 4b compare the results of X-test to the results published in [5].

	X-test			[5]		
circuit	weights	length	und	weights	length	und
c432	2	636	0	1	1 139	0
c499	1	1 125	0	1	542	0
c880	1	765	0	2	1 515	0
c1355	7	3 059	0	1	1 527	0
c1908	3	3 539	0	1	3 922	7
c2670	3	7 689	0	2	5 758	107
c3540	2	3 351	0	1	3 663	6
c5315	2	2 279	0	2	1 851	0
c6288	1	39	0	1	126	0
c7552	5	9 276	0	2	3 174	333

Table 4a: Weight set computation by X-test and by [5] ($k=512$) (combinational circuits)

The column "und" denotes the number of non-redundant faults which are not detected during weighted random pattern testing. In all the cases, X-test obtained a complete coverage of all detectable faults, and it leads also in some cases to significant reduction of the number of weight sets and the test length. For a better comparison of X-test and the method presented in [5], the weight set generation of X-test was stopped when the number of undetected faults from [5] was reached. Table 5 shows that in cases where the structural analysis method stuck at an incomplete fault coverage X-test can provide the same fault coverage by less patterns and less weights.

	X-test			[5]		
circuit	weights	length	und	weights	length	und
s208	2	1 931	0	3	1 003	0
s298	1	194	0	1	327	0
s344	1	181	0	1	305	0
s349	1	231	0	1	305	0
s382	1	270	0	1	468	0
s386	2	1 553	0	1	1 859	1
s420	3	1 423	0	2	1 802	36
s444	1	376	0	1	407	0
s510	1	961	0	1	968	0
s526	1	1 150	0	2	1 822	0
s641	2	463	0	3	2 182	1
s713	1	682	0	2	1 694	5
s820	2	3 699	0	2	4 493	5
s832	3	2 507	0	3	4 780	9
s838	5	6 468	0	3	2 834	107
s953	2	1 422	0	2	5 223	6
s1196	6	4 584	0	3	5 654	22
s1238	5	8 835	0	3	6 924	30
s1423	2	2 009	0	3	2 738	7
s1488	1	1 384	0	2	1 677	2
s1494	1	1 384	0	2	1 986	1
s5378	3	7 384	0	3	10 323	27
s9234	6	14 046	0	3	13 628	479
s13207	3	7 026	0	3	16 145	338
s15850	3	7 985	0	4	15 696	317
s35932	1	127	0	1	201	0
s38417	11	42 606	0	4	41 157	465
s38584	3	12 581	0	4	22 118	208

Table 4b: Weight set computation by X-test and by [5] (*k=512*) (sequential circuits)

	X-test			[5]		
circuit	weights	length	und	weights	length	und
c1908	1	2 940	4	1	3 922	7
c2670	1	5 487	42	2	5 758	107
c3540	1	3 333	4	1	3 663	6
c7552	1	4 002	324	2	3 174	333
s386	1	1 552	1	1	1 859	1
s420	1	1 151	24	2	1 802	36
s641	1	462	0	3	2 182	1
s713	1	682	0	2	1 694	5
s820	2	3 699	0	2	4 493	5
s832	1	2 111	8	3	4 780	9
s838	1	2 406	94	3	2 834	107
s953	2	1 422	0	2	5 223	6
s1196	3	4 303	11	3	5 654	22
s1238	1	5 910	21	3	6 924	30
s1423	1	2 007	3	3	2 738	7
s1488	1	1 384	0	2	1 677	2
s1494	1	1 384	0	2	1 986	1
s5378	2	7 383	1	3	10 323	27
s9234	1	8 165	185	3	13 628	479
s13207	1	6 569	89	3	16 145	338
s15850	1	6 776	59	4	15 696	317
s38417	1	23 793	416	4	41 157	465
s38584	1	11 668	109	4	22 118	208

Table 5: Number of weights and test length if fault coverage is restricted to [5] (*k=512*)

The next experiment shows the results of the computation of weight sets after the easy to detect faults are eliminated. The results obtained from [5] are based on the computation of weight sets from a deterministic test set. Only circuits are listed which contain hard to detect faults (table 6).

	X-test			[5]		
circuit	weights	length	und	weights	length	und
c2670	3	1 842	0	4	4 034	0
c7552	3	4 806	0	8	4 885	0
s420	1	64	0	2	1 825	0
s838	1	538	0	3	3 916	0
s953	1	435	0	2	3 870	0
s1196	3	2 213	0	5	6 424	0
s1238	3	2 344	0	5	8 559	0
s5378	2	1 374	0	4	11 005	0
s9234	6	13 079	0	8	16 391	0
s13207	3	2 381	0	4	15 781	0
s15850	4	3 251	0	4	14 767	0
s38417	8	11 722	0	6	30 663	0
s38584	3	1 784	0	5	19 973	0

Table 6: Weight set computation after pseudo-random pattern simulation (step A of figure 3) (*k=512)*

For real applications the rounding of the weights must be taken into account, too. In the next experiment, three linear feedback shift registers $x^{14}+x^5+x^3+x+1$, $x^{15}+x+1$ and $x^{16}+x^5+x^3+x^2+1$ as shown in figure 6 were used (table 7).

	exact weights			rounded weights		
circuit	weights	length	und	weights	length	und
c432	2	636	0	2	843	0
c499	1	1 125	0	1	704	0
c880	1	765	0	1	796	0
c1355	7	3 059	0	2	2 897	0
c1908	3	3 539	0	4	4 872	0
c2670	3	7 689	0	6	7 552	0
c3540	2	3 351	0	3	3 171	0
c5315	2	2 279	0	2	1 416	0
c6288	1	39	0	1	74	0
c7552	5	9 276	0	21	18 849	0
s208	2	1 931	0	2	1 229	0
s298	1	194	0	1	564	0
s344	1	181	0	1	131	0
s349	1	231	0	1	97	0
s382	1	270	0	1	326	0
s386	2	1 553	0	2	2 291	0
s420	3	1 423	0	3	1 468	0
s444	1	376	0	2	270	0
s510	1	961	0	1	1 219	0
s526	1	1 150	0	2	1 076	0
s641	2	463	0	1	728	0
s713	1	682	0	1	1 011	0
s820	2	3 699	0	3	1 910	0
s832	3	2 507	0	2	2 668	0
s838	5	6 468	0	4	4 132	0
s953	2	1 422	0	2	1 997	0
s1196	6	4 584	0	4	10 358	0
s1238	5	8 835	0	7	9 738	0
s1423	2	2 009	0	2	1 469	0
s1488	1	1 384	0	2	1 155	0
s1494	1	1 384	0	2	1 155	0
s5378	3	7 384	0	3	9 879	0
s9234	6	14 046	0	9	23 083	0
s13207	3	7 026	0	3	7 532	0
s15850	3	7 985	0	5	14 495	0
s35932	1	127	0	1	237	0
s38417	11	42 606	0	26	39 956	0
s38584	3	12 581	0	3	14 536	0

Table 7: Comparison of exact and rounded weights (*k = 512*)

Due to the indeterministic behavior and the applied heuristics during pattern generation, the results using rounded

weights are sometimes even better than the exact results. But in many cases, we need more patterns, and sometimes even more weight sets if the actual hardware application is considered during weight generation. Nevertheless, the results are still comparable to the weight computation techniques which do not consider the hardware implementation at all.

Conclusion

Weighted random pattern testing is a widely acknowledged technique for implementing a built-in or a built-off test scheme. The efficiency of weighted random pattern testing depends on both the number of weight sets and the number of weighted random patterns. A new method for weight computation was presented which often provides more efficient weights than previously known techniques. The method generates deterministic test sets with a maximum number of "Don't Care" bits which are used as a starting point for weight computation. The deterministic test sets are partitioned in order to minimize the number of conflicts between patterns during weight computation.
The method also takes into account that for actual applications the weights have to be rounded. Even in this case it may outperform state-of-the-art techniques which use exact probabilities.

References

1 V.K. Agarwal, E. Cerny: "Store and Generate Built-In Testing Approach", Proc. 11th Int. Symp. Fault-Tolerant Comput., 1981, pp. 35-40.

2 S.B. Akers, W. Jansz: "Test Set Embedding in Built-In Self-Test Environment", Proc. IEEE Int. Test Conf., Washington D.C., 1989, pp. 257-263.

3 M. Abramovici, P.R. Melon, D.T. Miller: "Critical Path Tracing - An Alternative to Fault Simulation", Proc. of 20th Design Automation Conf., 1983, pp. 214-220.

4 Z. Barzilai, D. Coppersmith, A.L. Rosenberg: "Exhaustive Generation of Bit Patterns with Applications to VLSI Self-Tesing", IEEE Trans. on Comp., Vol. C-32, No. 2, Feb. 1993, pp. 190-194.

5 M. Bershteyn: "Calculation of Multiple Sets of Weights for Weighted Random Testing", IEEE Int. Test Conf. 1993, pp.1031-1040.

6 F. Brglez et al.: "Accelerated ATPG and fault grading via testability analysis"; Proceedings IEEE Int. Symposium on Circuits and Systems, Kyoto, 1985.

7 F. Brglez, D. Bryan, K. Kozminski: "Combinational Profiles of Sequential Benchmark Circuits", Proc. IEEE Int. Symposium on Circuits and Systems, 1989, pp. 1929-1934.

8 F. Brglez et al: "Hardware-Based Weighted Random Pattern Generation for Boundary-Scan", Proc. IEEE Int. Test Conf., Washington D.C., 1989, pp. 264-274.

9 W. Daehn, J. Mucha: "Hardware Test Pattern Generators for Built-In Test", Proc. IEEE Int. Test Conf., 1981, pp. 110-113.

10 C. Dufaza, H. Viallon, C. Chevalier: "BIST Hardware Generator for Mixed Test Scheme", Proc. IEEE European Design and Test Conference, 1995, pp.424-430.

11 E.B. Eichelberger, E. Lindbloom: "Random-Pattern Coverage Enhancement and Diagnosis for LSSD Logic Self-Test", IBM J. Res. Develop., Vol. 27, No. 3, May 1983.

12 H. Fujiwara, T. Shimono: "On the Acceleration of Test Generation Algorithms", IEEE Transactions on Computers, Vol. C-32, No. 12, December 1983, pp. 1137-1144.

13 R. Kapur, S. Patil, T.J. Snethen, T.W. Williams: "Design of an Efficient Weighted Random Pattern Generation System", IEEE Int. Test Conf, 1994, pp.491-500.

14 B. Koenemann, J.Mucha, G. Zwiehoff: "Built-In Logik Block Observation Techniques", Proceedings Test Conf., Cherry Hill 1979, New Jersey.

15 R. Krieger, B. Becker, R. Sinkovic: "A BDD-based Algorithm for Computation of Exact Fault Detection Probabilities", 23rd Int. Symp. on Fault-Tolerant Computing, 1993, pp. 186-195.

16 R. Lisanke, F. Brglez, A.J. Degeus, D. Gregory: "Testability Driven Random Test-Pattern Generation", IEEE Trans. on Computer-Aided Design, Vol. CAD-6, No. 6, November 1987, pp. 1082-1087.

17 F. Muradali, V.K. Agarwal, B. Nadeau-Dostie: "A New Procedure for Weighted Random Built-In Self-Test", IEEE Int. Test Conf. 1990, pp. 660-669.

18 S. Pateras, J. Rajski: "Cube-Contained Random Patterns and their Application to the Complete Testing of Synthesized Multi-level Circuits", IEEE Int. Test Conf., 1991, pp.473-482.

19 I. Pomeranz, L.N. Reddy, S.M. Reddy: "COMPACTEST: A Method to Generate Compact Test Sets for Combinational Circuits", Proc. Int. Test Conf., 1991, pp. 194-203.

20 I. Pomeranz, S.M. Reddy: "3-Weight Pseudo-Random Test Generation Based on a Deterministic Test Set for Combinational and Sequential Circuits", IEEE Trans. on Computer-Aided Design, Vol. 12, No. 7, 1993, pp. 1050-1058.

21 J. Savir, P.H. Bardell: "On Random Pattern Test Length", IEEE Trans. on Comp. Vol. C-33, No. 6, 1984.

22 M. Schulz and E. Auth: "Advanced Automatic Test Generation and Redundancy Identification Techniques", Proc. 18th Int. Symp. Fault-Tolerant Comput., Tokyo 1988, pp. 30-35.

23 A.P. Ströle, H.-J. Wunderlich: "TESTCHIP: A Chip for Weighted Random Pattern Generation, Evaluation and Test Control", IEEE Journal of Solid-State Circuits, Vol. 26, No. 7, July 1991, pp. 1056-1063.

24 J.A. Waicukauski, E. Lindbloom, E.B. Eichelberger, O.P Forlenza: "A Method for Generating Weighted Random Test Patterns", IBM J. Res. Develop., Vol. 33, No. 2, March 1989, pp. 149-161.

25 L.T Wang, E.J. McCluskey: "Circuits for Pseudo-Exhaustive Test Pattern Generation", Proc. IEEE Int. Test Conf., 1986.

26 H.-J. Wunderlich: "PROTEST: A Tool for Probabilistic Testability Analysis", Proc. of the 22nd Design Automation Conf., 1985, pp. 204-211.

27 H.-J. Wunderlich: "Self Test Using Unequiprobable Random Patterns", Proc. 17th Int. Symp. Fault-Tolerant Comput., Pittsburgh 1987, pp. 258-263.

28 H.-J. Wunderlich: "On Computing Optimized Input Probabilities for Random Tests", 24th Design Automation Conf. 1987, pp.392-398.

29 H.-J. Wunderlich: "Multiple Distributions for Biased Random Test Patterns", IEEE Trans. on Computer-Aided Design, Vol. 9, No. 6, 1990, pp. 584-593.