On Computing Optimized Input Probabilities for Random Tests

Hans-Joachim Wunderlich

Universität Karlsruhe
Institut für Informatik IV
(Prof. Dr. D. Schmid)
7500 Karlsruhe
FRG
Phone: (0721) 608-4257
Mail: wunderlich%IRAVCL@germany.csnet

0. Abstract

Self testing of integrated circuits by random patterns has several technical and economical advantages. But there exists a large number of circuits which cannot be randomly tested, since the fault coverage achieved that way would be too low. In this paper we show that this problem can be solved by unequiprobable random patterns, and an efficient procedure is presented compu - ting the specific optimal probability for each primary input of a combinational network.

Those optimized random patterns can be produced on the chip during self test or off the chip in order to accelerate fault simu - lation and test pattern generation.

Keywords: Optimized random test, self test, fault detection probabilities, fault simulation.

1. Introduction

For application specific integrated circuits in small and medium sized charges the costs of the production test can reach more than 60% [Benn84] or even 70% [Will86] of the overall chip costs. One way to handle this problem may be testing by random pat - terns.

Here we can dispense with the time consuming automatic test pattern generation, and the application of those patterns needs no expensive test equipment, since it can be done by linear feedback shift registers (LFSR) during self test. This is possible in high speed, and therefore many technology dependent dynamic faults are detected in addition ([Tsai83], [WuRo86]).

Since a randomly generated test set is larger than a deterministic one, the detection rate of logical faults not in the fault model, multiple faults for instance, will be higher.

Now let δ be the confidence of a random test of length N, that is the probability to detect all faults $f \in F$ of the fault model F by applying N randomly generated patterns. If we assume that the detection of some faults by a pattern set of size N forms completely independent events, then we have

(1)

$$\delta = \prod_{f \in F} (1-(1-p_f)^N)$$

where p_f is the detection probability of the fault f.

For large N the assumption of independence is asymptotically fulfilled, but in general computing the necessary test length N by formula (1) will only provide an upper bound. Regarding correlated faults we would have to modify (1) slightly [Tyro86], but there is no efficient procedure known to compute those corre - lations. But for our purposes it is sufficient to compute an upper bound of the test length.

Furthermore we need not consider the bias of formula (1) due to pseudo-random testing [ChCI85], since for patterns with large bit width the difference is minimum. Thus formula (1) is precise enough, and it can be evaluated by tools computing fault detec - tion probabilities for combinational circuits.

For circuits with tree structures such algorithms were presented by P. and V. D. Agrawal [AgAg75], and the general case was solved by Parker and McCluskey [McPa75]. But the latter procedure shows an exponential time and storage complexity due to the NP-completeness of the underlying fault detection pro - blem. Therefore in recent years much work was done to *estimate* those probabilities.

The cutting algorithm [BDS84] determines upper and lower bounds for the probabilities, PROTEST ([Wu84], [Wu85]), and a new version of PREDICT [ABS86] estimate by an analysing procedure, and STAFAN [AgJa84] uses counting of signal values during simulation.

But now it turns out that there are many circuits which cannot be randomly tested due to faults with low detection probabilitites. Table 1 shows, for some circuits, the necessary test lengths based on the estimations of PROTEST.

This research was supported by the BMFT (Bundesministerium für Forschung und Technologie) of the Federal Republic of Germany under grant NT 2809 A 3

	Circuit	Required test length
*	S1	$5.6*10^8$
*	S2	$2.0*10^{11}$
	C432	$2.5*10^3$
	C499	$1.9*10^3$
	C880	$3.7*10^4$
	C1355	$2.2*10^6$
	C1908	$6.2*10^4$
*	C2670	$1.1*10^7$
	C3540	$2.3*10^6$
	C5315	$5.3*10^4$
	C6288	$1.9*10^3$
*	C7552	$4.9*10^{11}$

Table 1: Necessary test lengths for a conventional random test (by PROTEST).

The circuits C<n> are the well known benchmarks of the ISCAS'85 test session [BRGL85], the circuit S1 is a 24-bit comparator constructed by six Texas Instruments comparators SN 7485 [TI80], where some redundancies are removed, and S2 is the combinational part of a 32 bit divider [KuWu85].

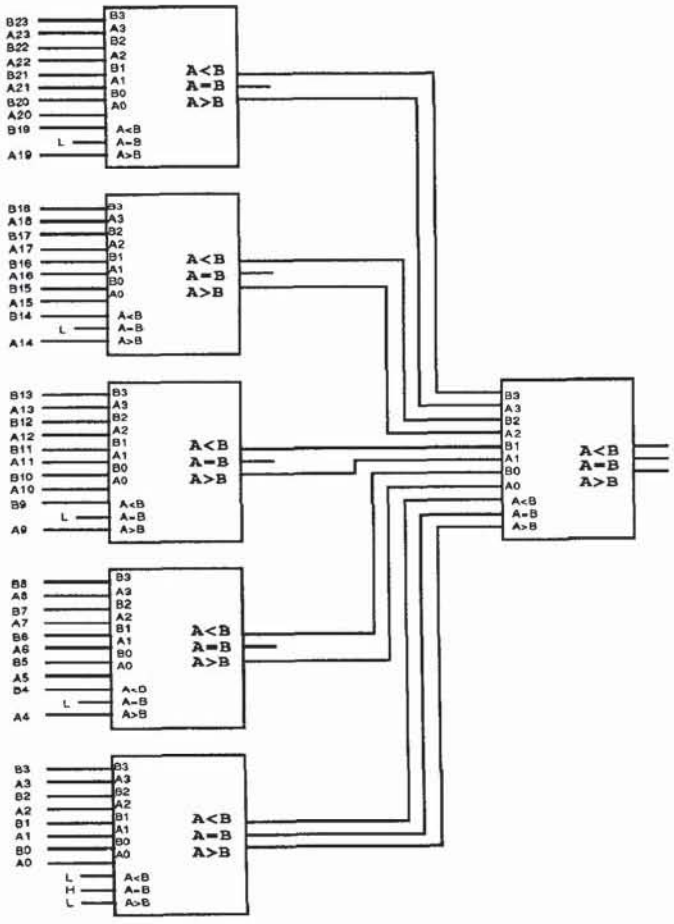

Fig. 1: 24-bit comparator S1

In table 1 the marked (*) circuits need an exorbitant size of the random test set, and those predictions of PROTEST are confirmed by fault simulation:

	Circuit	Test length	Fault coverage
*	S1	12,000	80.7 %
*	S2	12,000	77.2 %
*	C2670	4,000	88.0 %
*	C7552	4,096	93.9 %

Table2: Fault coverage by simulation of conventional random patterns.

It should be noted that an estimation with the exact value 0 or 1 of a signal probability by PROTEST is a proof (not an estimation!) of redundancy. But of course not in all cases a fixed signal value can be detected this way, and therefore there may be redundancies left which cannot be found by PROTEST.

The fault coverage in table 2 is computed only with respect to those faults which are not proven to be undetectable due to redundancy. This explains the difference to the results of Carter et al. [CART85], where the fault coverage was even lower. The table indicates that self testing of those circuits may need several hours, preventing an economical use of random patterns.

But PROTEST suggests also specific probabilities to set each primary input to logical "1". Using such optimized input probabilities PROTEST proposes the test lengths listed in table 3:

	Circuit	Required test length
*	S1	$1.5*10^4$
*	S2	$4.0*10^4$
*	C2670	$6.9*10^4$
*	C7552	$1.2*10^5$

Table 3: Necessary test lengths for optimized random tests (by PROTEST).

The results of fault simulation listed in table 4 prove that such optimized random patterns yield a higher fault coverage indeed. A suspicious reader may verify this by random patterns generated with respect to the optimized input probabilities listed in the appendix.

	Circuit	Test length	Fault coverage
*	S1	12,000	99.7 %
*	S2	12,000	99.7 %
*	C2670	4,000	99.7 %
*	C7552	4,000	98.9 %

Table 4: Fault coverage by simulation of optimized random patterns

For the rest of this paper we are dealing with the optimizing problem. In section 2 we introduce an objective function for

input probabilities, and in section 3 we discuss its mathematical properties. In section 4 we describe the implemented optimizing procedure. Some applications, the limits of the approach, and the performance are discussed in section 5.

The examination are based on the assumption that there is a tool available computing or estimating fault detection probabilities efficiently. For the reported results the estimation procedures of PROTEST have been used, but with slight modifications PREDICT or STAFAN will presumably work as well.

2. An objective function for input probabilities

2.1 Some definitions

The most widely used self test techniques configure the circuit registers to linear feedback shift registers in order to produce and to evaluate test patterns. Therefore we can restrict our exami - nations to combinational networks.

A combinational network C has nodes $K := \langle k_1,\ldots,k_r\rangle$, some special nodes $I := \langle i_1,\ldots,i_s\rangle$, the primary inputs, and some special nodes $O := \langle o_1,\ldots,o_t\rangle$, the primary outputs. We define an input variable x of a combinational network C as a boolean random variable, and P(x) is the probability that x is true. The tupel $X := \langle x_i | i \in I\rangle$ defines for each primary input an input variable, we assume that those variables are completely indepen - dent.

For three boolean random variables we have

(2) $P(\neg x) = 1-P(x)$

(3) $P(x\&y\&z) =$ P(x)P(y)P(z), if x, y and z are independent; P(y&z), if x=y; else additional informations are required.

The set of boolean functions

$$\{f^b:\{\text{TRUE, FALSE}\}^n \rightarrow \{\text{TRUE, FALSE}\} \mid n\in N\}$$

is isomorphically mapped into the set of arithmetical functions

$$\{f^a:\{0,1\}^n \rightarrow \{0,1\} \mid n\in N\}$$

by the following rules:

(4)

TRUE	$\rightarrow$	1
FALSE	$\rightarrow$	0
$x^b\&y^b$	$\rightarrow$	x^*y
$\neg x^b$	$\rightarrow$	1-x

Let $x_1,\ldots,x_n$ be boolean random variables with $P(x_i) =: p_i$ and let $y_1,\ldots,y_n$ be two-valued arithmetical variables from $\{0,1\}$ with $P(y_i=1) = p_i$. Then the expectation values are $E(y_i) = p_i$, and the probability of a boolean function being true is equal to the expectation value of the corresponding arithmetical function:

(5) $$P(f^b(x_1,\ldots,x_n)) = E(f^a(y_1,\ldots,y_n)).$$

An arithmetical embedding of a boolean function

$$f^b:\{\text{TRUE, FALSE}\}^n \rightarrow \{\text{TRUE, FALSE}\}$$

is the real function $f: [0,1]^n \rightarrow [0,1]$, defined by

(6) $$f(z_1,\ldots,z_n) := P(f^b(x_1,\ldots,x_n)).$$

where the completeley independent boolean random variables have the probability $P(x_i) = z_i$.

Notation: Let $Z := \langle z_1\ldots z_n\rangle$, and $y \in [0,1]$. We write $f(Z,y|_i) := f(z_1,\ldots,z_{i-1},y,z_{i+1},\ldots,z_n)$.

Lemma 1: For each arithmetical embedding f we have

(7) $$f(Z) = z_i f(Z,1|_i) + (1-z_i)f(Z,0|_i)$$

Proof: By the Shannon expansion.

The input variables x_i determine the boolean random variables x_k for all nodes $k\in K$ with the signal probabilities $P(x_k)$. The input probabilities are the signal probabilities at the primary inputs.

Let f be an arbitrary fault changing a gate function into another combinational function, and let X be a tupel of input variables. The fault detection probability $p_f(X)$ of f is the probability that f is detected by a random pattern generated according to the distributions of X.

2.2 Preliminary work by other authors

P. and V. D. Agrawal proposed an algorithm which computes in - put probabilities of reconvergent free networks maximizing the probability of path sentizing [AgAg75a]. For the general case Agrawal and Seth tried to optimize input probabilities by infor - mation theoretic means ([Agra81], [AgSe82]), which has the disadvantage that the real fault model and fault coverage are not directly involved. Lieberherr compared two optimizing ap - proaches [Lieb84]: On the one hand optimizing path sen - sitization, on the other hand the generation of patterns setting al - ways k inputs to logical "1", and finding an optimal k. He didn't present optimizing procedures.

In the following we define a new objective function based on the real circuit structure and on the real fault model.

2.3 The definition of the objective function

Throughout the paper we assume an arbitrary but fixed combi - national fault model F. This is adequate for the large number of circuits in bipolar technologies, in nMOS pull-down designs, in dynamic nMOS and domino CMOS [WuRo86]. However the treatment of sequential st-open faults in static CMOS and nMOS pass transistor designs would require some modifications. F may contain an arbitrary number of such faults, and it must contain all stuck-at-0 and stuck-at-1 faults at the primary inputs. Further - more all faults of F must be detectable.

For each fault $f\in F$ the detection probability $p_f(X)$ depends on the tupel of input probabilities $X := \langle x_i | i\in I\rangle$. Therefore formula (1) turns into

(8)

$$\delta_N(X) = \prod_{f \in F}(1-(1-p_f(X))^N).$$

This formula expresses the probability that all faults are detected by N patterns with the distributions of X. Using some well known approximations (8) is transformed into

(9)

$$\ln(\delta_N(X)) \approx -\sum_{f \in F}(1-p_f(X))^N \approx -\sum_{f \in F} e^{-Np_f(X)}$$

Formula (9) describes our objective function and we call a tupel X of input probabilities optimal with respect to N, if

(10)

$$J_N(X) := \sum_{f \in F} e^{-Np_f(X)}$$

is minimum at $X \in [0,1]^I$

3. Mathematical properties of the objective function

3.1 Classification of the optimizing problem

In the {0,1}-space expectation value and probability coincide, and the stochastical optimizing problem reduces to a deterministic one. But this is only a modest simplification, since one immediately notices that the objective function is not a member of the well known linear or quadratical optimizing problems.

Examining only the stuck-at faults at the primary inputs $A_0,\ldots,A_{23}$, $B_0,\ldots,B_{23}$ of circuit S1 one can easily verify that the objective function will have at least 2^{24} minimum points. Thus in general the objective function will not be convex or even unimodal. Our optimizing problem is a member of the general class of smooth multi-extremal problems, which have an exponential average case complexity with respect to the number of variables, and to the required precision [NeYu83].

Furthermore the known global optimizing procedures like the Newton or the gradient method will fail to handle large circuits with hundreds or thousands of input variables resulting from scan designs. Therefore we don't try to find a global optimum, but we use some approximations to search a relative one. Here the fundamental means are provided by the next section.

3.2 Optimization with respect to one variable

We will show that the objective function is strictly convex with respect to one single variable. Hence for each fixed $x_1,\ldots,x_{i-1}, x_{i+1},\ldots,x_n$ there exists exactly one $x_i \in [0,1]$ with minimum $J_N(x_1,\ldots,x_i,\ldots x_n)$. First we define again $J_N(X,y|_i) := J_N(x_1\ldots,x_{i-1},y_i,x_{i+1},\ldots x_n)$. Then we observe:

Lemma 2: For all $X \in]0,1[^I$ and all sufficiently large N we have

(11)

$$J_N(X,0|_i) > J_N(X) < J_N(X,1|_i).$$

Proof: If N is sufficiently large formula (9) yields $1 \approx \delta_N(X) \approx \exp(-J_N(X))$, and thus $J_N(X) \approx 0$. But if for instance $x_i = 1$, then the stuck-at-1 fault at x_i is not detectable, has detection probability 0, and thus $J_N(X) \geq 1$.

Now we use a well known convexity criterium:

Lemma 3: For all $i \in I$ we have

(12)

$$\frac{d^2 J_N(X,y|_i)}{d^2 y} > 0$$

and therefore $J_N(X,y|_i)$ is strictly convex in y.

Proof: Using Lemma 1 we have

$$p_f(X) = p_f(X,0|_i) + x_i(p_f(X,1|_i)-p_f(X,0|_i)).$$

Hence

(13)

$$\frac{dJ_N(X,y|_i)}{dy} = \sum_{f \in F} -N(p_f(X,1|_i)-p_f(X,0|_i))e^{-Np_f(X,y|_i)}$$

and

(14)

$$\frac{d^2 J_N(X,y|_i)}{d^2 y} \sum_{f \in F} N^2(p_f(X,1|_i)-p_f(X,0|_i))^2 e^{-Np_f(X,y|_i)} > 0.$$

The > holds since we assume irredundancy, and at some primary input we have $p_f(X,1|_i)-p_f(X,0|_i) \neq 0$.

And no we have by Lemma 2 and Lemma 3:

> For each $X \in [0,1]^I$ there exists exactly one $y \in [0,1]$ with $dJ_N(X,y|_i)/dy = 0$ and $J_N(X,y|_i)$ has there its minimum.

This minimum point can be computed by a simple iteration:

(15)

$$y^+ := y - \frac{\frac{dJ_N(X,y|_i)}{dy}}{\frac{d^2 J_N(X,y|_i)}{d^2 y}}$$

4. The optimizing procedure

In this section we discuss some implemented procedu-res mini - mizing the objective function. First we observe two facts with very important impact to the efficiency:

(1) Already Bardell and Savir [BaSi84] noticed that only the hardest detectable faults are relevant to the necessary test length N. If N satisfies formula (7), we have $J_N(X) = \beta \approx 0$, and for instance, if there are two faults f and g with $p_f = 10*p_g$, then $\exp(-Np_g) + \exp(-N*10p_g) \leq \beta$ holds. Thus $\exp(-Np_g) \leq \beta$ and $\exp(-Np_f) = \exp(-N*10p_g) = \exp(-Np_g)^{10} \leq \beta^{10}$.

Therefore p_f doesn't contribute any numerically computable value to the objective function. Hence during one optimizing step we have only to deal with the small subset F' of the hardest detectable faults of F. (But caution! The order of the detection proba - bilities may change during optimization).

(2) Formula (15) needs the values of $p_f(X,0_i)$ and $p_f(X,1_i)$, $f \in F'$. They can be computed before the iteration by an effort which is less than twice of the testability analysis. Thus the minimizing procedure itself is nearly independent of the circuit size!

Now we discuss the implemented procedures in deeper detail:

SORT (F):

Output is the sorted fault list $f_1 \ldots f_n$ in the order of increasing probabilities, where n is the total number of faults, and all known redundancies are removed.

NORMALIZE(N,nf):

If a sorted fault list is given the procedure computes the minimum number N of random patterns to satisfy (7), nf will be the number of faults with low detection probabilities, that is $F' = \{f_1 \ldots f_{nf}\}$. Roughly the algorithm is like this:

Set $Q := \ln(\delta)$, and define the function

$$l(z,M) := \sum_{1 \leq i \leq z} e^{-p_{fi}M}$$

which is a lower bound of $J_M(X)$. Set

$$u(z,M) := l(z,M) + (n-z)e^{-p_{fz}M}$$

which is an upper bound of $J_M(X)$. We already remarked that for fixed M only few z are needed to check $l(z,M) > Q$ or $u(z,M) < Q$. In the first case the desired N must be larger than M, in the second case we have $N < M$. Hence using intervall sections we find an N and an integer z with $u(z,N-1) < Q$ and $l(z,N) > Q$. Now we set nf := z.

ANALYSIS (X,F):

Using the input probabilities X the list F of detection probabilities is computed by PROTEST.

PREPARE (X,i,nf,F,F_0_1):

```
Input:   X    {Input probabilities}
         i    {Input number to be optimized}
         nf   {Number of relevant faults}
         F    {Sorted fault list};

Output:  F_0_1  {List of pf(X,0i) and pf(X,1i) for all f∈F}

Begin
         YO := (X,0i);
         F_0 := F;
         ANALYSIS (Y0, F_0);
         Y1 := (X,1i);
         F_1 := F;
         ANALYSIS (Y1, F_1);
         F_0_1 :=  Ordered list with the corres-
                   ponding pairs of F_0 and F_1.
end;
```

MINIMIZE (F_0_1,N,Y):

This procedure implements the iteration of formula (15). Y will be the minimizing value for x_i.

OPTIMIZE:

```
Begin
    X := Starting vector;
    ANALYSIS (X,F);
    SORT(F)
    NORMALIZE(N_new,nf)
    N_old := the maximum possible value of N;
    While (N_old - N_new) > σ do
                          {σ is user definded}
    Begin
        N_old := N_new;
        For i=1 to INP do
        Begin
            PREPARE (X,i,nf,F,F_0_1);
            MINIMIZE (F_0_1,N_new,y);
            xi := y;
            ANALYSIS(X,F);
            SORT(F);
            NORMALIZE(N_new,nf);
        end;
    end;
end;
```

In the next section we discuss the performance and some appli - cations.

5. Practical experience

5.1 Performance

During the optimization of an primary input i the ANALYSIS procedure is called three times, but each time with the same values X except for x_i. If ANALYSIS takes this into account then optimizing one input variable will take less effort than a complete testability analysis in most cases.

Table 5 lists the performance statistics for the circuits mentioned in section 1. The results are achieved by a SIEMENS 7561 computer, a machine with approximately 2.5 MIPS.

	Circuit	CPU time (sec)
*	S1	300
*	S2	600
*	C2670	1.200
*	C7552	2.000

Table 5: CPU time for optimizing input probabilities by PROTEST

5.2 Simulation

Self test by random patterns is the main goal of the optimizing approach. A self test modul similar to the well known BILBO is presented in [Wu86] and [Wu87].

But the optimizing procedure can also support deterministic test pattern generation, since the computing time of optimizing and simulation together is less than computing test patterns by the D-algorithm. Fault simulation of optimized patterns can provide nearly complete fault coverage in economical time. Fig. 2 illustrates the increase of fault coverage for optimized and conventional random patterns.

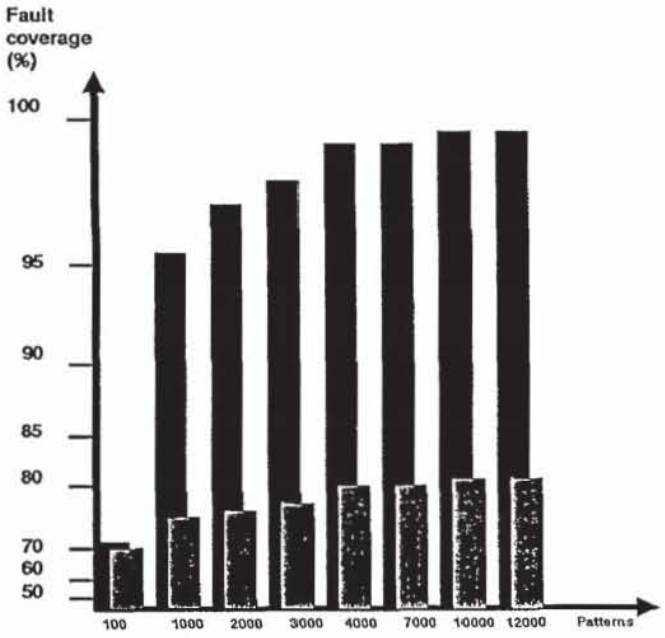

Fig.2: Fault coverage vs. pattern count (S1)

5.3 The limits of the approach

Up to now all examined circuits could be made random pattern testable by optimizing. For all circuits by the input probabilities that could be found, an optimized random self test needs less than 1 sec. test time. But of course circuits can be constructed, which cannot be processed by optimization. This is the case if there are pairs of faults with the following two properties:

- each of the faults has a very low detection probability, and
- the Hamming distance between the test sets of these both faults is very large.

This situation prevents the successfull optimizing for both faults simultaneously. The problem can be solved by partitioning the fault set, and by computing different optimal input probabilities for each part. But until now such pathological circuits didn't occur, and thus the additional procedure wasn't implemented yet.

6. Conclusion

Using optimized, unequiprobable random patterns the fault coverage can increase and the necessary test length can decrease by orders of magnitude. Hence the class of random testable circuits is enlarged distinctly this way.

Based on tools estimating fault detection probabilities an efficient procedure was presented, which computes for each primary input its optimal input probability.

The optimized random patterns can be applied during self test or during fault simulation.

Literature:

[AgAg75] Agrawal, P.; Agrawal, V.D.: Probabilistic Analysis of Random Test Generation Method for Irredundant Combinational Logic Networks; in: IEEE, Trans. on Comp., Vol. C-24, No. 7, July 1975

[AgAg75a] Agrawal, V.D.; Agrawal, P.: On Improving the Efficiency of Monte Carlo Test Generation; in: FTCS 5, 1975

[Agra81] Agrawal, V.D.: An Information Theoretic Approach to Digital Fault Testing; in: IEEE, Trans. Comp., Vol. C-30, No. 8, August 1981

[AgJa84] Jain, S.K.; Agrawal, V.D.: STAFAN: An Alternative to Fault Simulation; in: Proc. 21st Design Automation Conference, 1984

[AgSe92] Seth, S.C.; Agrawal, V.D.: Statistical Design Verification; in: Proc. FTCS-12, 1982

[ABS86] Seth, S.C. et al.: An Exact Analysis for Efficient Computation of Random-Pattern Testability in Combinational Circuits; in: FTCS 16, 1986

[BaSi84] Savir, J.; Bardell, P.H.: On Random Pattern Test Length; in: IEEE, Trans. on Comp., Vol. C-33, No. 6, June 1984

[Benn84] Bennetts, R.G.: Design of Testable Logic Circuits, Addison-Wesley, 1984

[BDS84] Savir,J.: Random Pattern Testability; in: IEEE, Trans. on Comp., Vol. C-33, No. 1, 1984

[BRGL85] Brglez, F. et al.: Accelerated ATPG and fault grading via testability analysis; in: Proc. ISCAS´85, Kyoto 1985

[ChCl85] Chin, C.K.; McCluskey, E.J.: Test Length for Pseudo Random Testing; in: Proc. International Test Conference, 1985

[CART85] Carter, J.L. et al.: ATPG via Random Pattern Simulation; in: Proc. ISCAS´85, Kyoto 1985

[KuWu85] Kunzmann, A.; Wunderlich, H.-J.: Design automation of random testable circuits; in: Proc. ESSCIRC'85, Toulouse 1985

[Lieb84] Lieberherr, K.J.: Parameterized Random Testing; in: Proc. 21st Design Automation Conference, 1984

[McPa75] Parker, K.P.; McCluskey, E.J.: Analysis of Logic Circuits with Faults Using Input Signal Probabilities; in: IEEE, Trans. on Comp., Vol. C-24, No. 5, May 1975

[NeYu83] Nemirovsky, A.S.; Yudin, D.B.: Problem Complexity and Method Efficiency in Optimization; John Wiley & Sons, 1983

[Tyro86] Tyron, D.R.: Self-Testing with Correlated Faults; in: Proc. 23rd Design Automation Conference, Las Vegas 1986

[Tsai83] Tsai, M.Y.: Pass Transistor Networks in MOS Technology: Synthesis, Performance, and Testing; in: Proc. IEEE Int. Symp. of Circuits and Systems, 1983

[TI80] The TTL Data Book; Texas Instruments, 1980

[Will86] Williams, R.M.: "IBM Perspectives on the Electrical Design Automation Industry", Keynote Address to the 23rd Design Automation Conference, Las Vegas 1986

[Wu84] Wunderlich, H.-J.: Zur statistischen Analyse der Testbarkeit digitaler Schaltungen; Universität Karlsruhe, Fakultät für Informatik, Interner Bericht 18/84, 1984

[Wu85] Wunderlich, H.-J.: PROTEST: A Tool for Probabilistic Testability Analysis; in: Proc. 22nd Design Automation Conference, Las Vegas, 1985

[Wu86] Wunderlich, H.-J.: Probabilistische Verfahren zur Verbesserung der Testbarkeit synthetisierter digitaler Schaltungen; Dissertation an der Fakultät für Informatik der Universität Karlsruhe, 1986

[Wu87] Wunderlich, H.-J.: Self test using unequiprobable random patterns; submitted

[WuRo86] Wunderlich, H.-J.; Rosenstiel, W.: On Fault Modeling for Dynamic MOS Circuits; in: Proc. 23rd Design Automation Conference, Las Vegas 1986

Appendix

Optimized input probabilities for the circuit C2670

Inputs	Probability
1-3	0.5
4	0.1
5-6	0.15
7	0.5
8	0.7
9	0.9
10-11	0.5
12	0.55
13-20	0.1
21	0.15
22	0.9
23	0.55
24	0.1
25-26	0.15
27	0.2
28	0.5
29	0.05
30	0.85
31	0.05
32	0.5
33-36	0.05
37	0.1
38	0.05
39	0.1
40	0.15
41	0.5
42	0.05
43	0.5
44	0.05
45-46	0.25
47	0.15
48	0.05
49-50	0.3
51	0.5
52	0.05
53	0.5
54	0.15
55	0.1
56-61	0.05
62	0.5
63	0.15
64	0.5
65	0.35
66	0.4
67	0.05
68	0.45
69	0.35
70	0.25
71	0.3
72	0.4
73-74	0.5
75	0.05
76	0.5
77	0.95
78-83	0.05
84	0.95
85	0.25
86	0.5
87	0.95
88	0.1
89-91	0.25
92	0.2
93	0.15
94	0.95
95	0.05
96	0.5
97	0.85
98	0.05
99-100	0.1
101	0.25
102	0.45
103	0.7
104	0.75
105	0.05
106	0.5
107	0.65
108-112	0.05
113	0.3
114	0.9
115-195	0.5
196	0.3
197-200	0.5
201-202	0.75
203	0.4
204-208	0.8
209	0.85
210-212	0.8
213	0.5
214	0.8
215-218	0.85
219	0.9
220	0.15
221	0.95
222	0.25
223-233	0.5

Optimized input probabilities for the circuit C7552

Inputs	Probability
1-2	0.5
3	0.7
4	0.95
5	0.5
6	0.75
7	0.9
8	0.95
9	0.85
10	0.9
11	0.95
12	0.1
13	0.05
14	0.9
15	0.1
16	0.9
17-18	0.05
19	0.15
20	0.25
21	0.5
22	0.95
23-26	0.1
27	0.95
28	0.85
29	0.1
30	0.9
31	0.95
32	0.7
33	0.3
34	0.5
35	0.45
36	0.35
37	0.05
38	0.15
39	0.05
40-41	0.85
42	0.05
43	0.1
44	0.85
45	0.7
46-48	0.95
49	0.65
50	0.85
51	0.9
52	0.95
53	0.85
54	0.05
55	0.95
56	0.9
57-59	0.95
60	0.05
61	0.2
62	0.9
63	0.1
64	0.95
65	0.85
66	0.7
67-68	0.5
69	0.15
70	0.85
71	0.95
72	0.35
73	0.8
74-76	0.95
77-83	0.05
84	0.1
85	0.05
86	0.95
87-89	0.5
90-93	0.05
94	0.7
95-96	0.95
97-100	0.05
101	0.1
102-103	0.05
104-109	0.95
110	0.7
111	0.95
112	0.9
113-116	0.95
117-118	0.05
119-120	0.95
121	0.05
122	0.95
123	0.75
124-131	0.95
132	0.05
133	0.6
134	0.95
135-136	0.5
137-140	0.05
141-146	0.95
147-148	0.05
149-150	0.95
151	0.1
152-163	0.95
154	0.7
155	0.95
156	0.85
157	0.05
158-160	0.95
161	0.9
162-163	0.95
164	0.7
165-166	0.5
167	0.95
168	0.75
169-172	0.05
173	0.1
174	0.15
175-176	0.95
177	0.85
178	0.8
179-181	0.05
182-184	0.1
185-186	0.05
187	0.1
188-191	0.95
192	0.65
193	0.6
194	0.25
195	0.9
196	0.1
197	0.9
198-199	0.95
200-201	0.05
202	0.9
203-205	0.95
206-207	0.5