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@ira.uka.de

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 computing 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 simulation 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% [Ben84] or even 70% [Wil86] of the overall chip costs. One way to handle this problem may be testing by random patterns.

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], [Wu86]).

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 \( \delta \) 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

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

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 [Tyr06], but there is no efficient procedure known to compute those correlations. 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 [ChCh85], 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 detection 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 problem. 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 analyzing 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 probabilities. 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

24th ACM/IEEE Design Automation Conference

Paper 24.2

392

© 1987 ACM 0738-100X/87/0600-0392$00.75
The circuits C<nt> 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 SN7485 [T80], where some redundancies are removed, and S2 is the combinational part of a 32-bit divider [KuWu85].

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:

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Required test length</th>
</tr>
</thead>
<tbody>
<tr>
<td>* S1</td>
<td>5.6*10^3</td>
</tr>
<tr>
<td>* S2</td>
<td>2.0*10^4</td>
</tr>
<tr>
<td>C432</td>
<td>2.5*10^3</td>
</tr>
<tr>
<td>C499</td>
<td>1.9*10^3</td>
</tr>
<tr>
<td>C800</td>
<td>3.7*10^4</td>
</tr>
<tr>
<td>C1355</td>
<td>2.2*10^4</td>
</tr>
<tr>
<td>C1908</td>
<td>6.2*10^4</td>
</tr>
<tr>
<td>* C2670</td>
<td>1.1*10^1</td>
</tr>
<tr>
<td>* C7552</td>
<td>4.9*10^1</td>
</tr>
<tr>
<td>C3540</td>
<td>2.3*10^4</td>
</tr>
<tr>
<td>C5315</td>
<td>5.3*10^4</td>
</tr>
<tr>
<td>C6288</td>
<td>1.9*10^3</td>
</tr>
<tr>
<td>* C7552</td>
<td>4.9*10^1</td>
</tr>
</tbody>
</table>

Table 1: Necessary test lengths for a conventional random test (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:

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Required test length</th>
</tr>
</thead>
<tbody>
<tr>
<td>* S1</td>
<td>12,000</td>
</tr>
<tr>
<td>* S2</td>
<td>12,000</td>
</tr>
<tr>
<td>* C2670</td>
<td>4,000</td>
</tr>
<tr>
<td>* C7552</td>
<td>4,096</td>
</tr>
</tbody>
</table>

Table 2: Fault coverage by simulation of conventional random patterns.

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:

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Test length</th>
<th>Fault coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>* S1</td>
<td>12,000</td>
<td>80.7%</td>
</tr>
<tr>
<td>* S2</td>
<td>12,000</td>
<td>77.2%</td>
</tr>
<tr>
<td>* C2670</td>
<td>4,000</td>
<td>88.0%</td>
</tr>
<tr>
<td>* C7552</td>
<td>4,096</td>
<td>93.9%</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Test length</th>
<th>Fault coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>* S1</td>
<td>12,000</td>
<td>99.7%</td>
</tr>
<tr>
<td>* S2</td>
<td>12,000</td>
<td>99.7%</td>
</tr>
<tr>
<td>* C2670</td>
<td>4,000</td>
<td>99.7%</td>
</tr>
<tr>
<td>* C7552</td>
<td>4,096</td>
<td>98.9%</td>
</tr>
</tbody>
</table>

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 examinations to combinational networks.

A combinational network \( C \) has nodes \( K := \langle k_1, \ldots, k_l \rangle \), some special nodes \( I := \langle i_1, \ldots, i_m \rangle \), the primary inputs, and some special nodes \( 0 := \langle o_1, \ldots, o_n \rangle \), 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 tuple \( X := \langle x_i | i \in I \rangle \) defines for each primary input an input variable, we assume that those variables are completely independent.

For three boolean random variables we have

\[
\begin{align*}
P(\neg x) &= 1 - P(x) \\
P(x \land y \land z) &= P(x)P(y)P(z), \text{ if } x, y \text{ and } z \text{ are independent; } \\
P(x \land y \land z) &= P(y \land z), \text{ if } x = y; \\
&\text{else additional informations are required.}
\end{align*}
\]

The set of boolean functions

\[
\{ b : \{ \text{TRUE, FALSE} \}^n \rightarrow \{ \text{TRUE, FALSE} \} \mid n \in \mathbb{N} \}
\]

is isomorphically mapped into the set of arithmetical functions

\[
\{ b : \{0,1\}^n \rightarrow \{0,1\} \mid n \in \mathbb{N} \}
\]

by the following rules:

\[
\begin{align*}
\text{TRUE} &\rightarrow 1 \\
\text{FALSE} &\rightarrow 0 \\
x^b &\rightarrow x^y \\
\neg x^b &\rightarrow 1 - x
\end{align*}
\]

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:

\[
P(b(x_1, \ldots, x_n)) = E(b(y_1, \ldots, y_n)).
\]

An arithmetical embedding of a boolean function

\[
b : \{ \text{TRUE, FALSE} \}^n \rightarrow \{ \text{TRUE, FALSE} \}
\]

is the real function \( f : \{0,1\}^n \rightarrow \{0,1\} \), defined by

\[
f(z_1, \ldots, z_n) := P(b(x_1, \ldots, x_n)).
\]

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: For each arithmetical embedding \( f \) we have

\[
f(Z) = z_if(Z,1) + (1-z_i)f(Z,0)
\]

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 \( p(X) \) 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 input probabilities of reconvergent free networks maximizing the probability of path sensitizing \([\text{AgAg85a}]\). For the general case Agrawal and Seth tried to optimize input probabilities by information theoretic means \([\text{AgRa81}, \text{AgSe82}]\), which has the disadvantage that the real fault model and fault coverage are not directly involved. Lieberherr compared two optimizing approaches \([\text{Lieb84}]\). On the one hand optimizing path sensitization, on the other hand the generation of patterns setting always \( k \) 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 combinational 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 \([\text{WuRo86}]\). 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. Furthermore 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
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

\[
\ln(\delta_N(X)) = -\sum_{f \in F} (1 - p_f(X))^N = -\sum_{f \in F} e^{-Np_f(X)}
\]

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

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

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 stochastic 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 quadratic optimizing problems.

Examining only the stuck-at faults at the primary inputs \( A_0, \ldots, A_{23}, B_0, \ldots, B_{23} \) of circuit \( S_1 \) 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 \( J_N(X, y_1) := J_N(x_1, \ldots, x_{i-1}, y_1, x_{i+1}, \ldots, x_n) \). Then we observe:

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

\[
J_N(X, 0|_I) > J_N(X) < J_N(X, 1|_I).
\]

**Proof:** If \( N \) is sufficiently large formula (9) yields

\[
1 = \delta_N(X) = \exp(-J_N(X)), \quad \text{and thus} \quad J_N(X) = 0.
\]

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

\[
\frac{d^2 J_N(X, y_1)}{dy^2} > 0
\]

and therefore \( J_N(X, y_1) \) 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

\[
\frac{d J_N(X, y_1)}{dy} = \sum_{f \in F} e^{-Np_f(X)}
\]

and

\[
\frac{d^2 J_N(X, y_1)}{dy^2} = \sum_{f \in F} \left( p_f(X, 1|_I) - p_f(X, 0|_I) \right)^2 e^{-Np_f(X)} > 0.
\]

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 now we have by Lemma 2 and Lemma 3:

For each \( X \in [0,1]^I \) there exists exactly one \( y \in [0,1] \) with \( dJ_N(X, y_1)/dy = 0 \) and \( J_N(X, y_1) \) has there its minimum.

This minimum point can be computed by a simple iteration:

\[
y^* := y - \frac{d J_N(X, y_1)}{dy} \frac{d^2 J_N(X, y_1)}{dy^2}^{-1}
\]

Paper 24.2

395
4. The optimizing procedure

In this section we discuss some implemented procedures minimizing the objective function. First we observe two facts with very important impact to the efficiency:

1. Already Bardell and Savir [BaSa84] noticed that only the hardest detectable faults are relevant to the necessary test length N. If N satisfies formula (7), we have $J(N) = \beta = 0$, and for instance, if there are two faults f and g with $p_f = 10^* p_g$, then $\exp(-N p_f) + \exp(-N*10p_g) \leq \beta$ holds. Thus $\exp(-N p_f) \approx 0$ and $\exp(-N p_g) = \exp(-N*10p_g) / \beta 10$. 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 probabilities may change during optimization).

2. Formula (15) needs the values of $p_r(X,O_i)$ and $p_f(X,1_j)$. 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 $f_1,...,f_n$ 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 $F = \{f_1,...,f_n\}$. Roughly the algorithm is like this:

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

$$I(z,M) := \sum_{1 \leq i \leq z} p_i^M$$

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

$$u(z,M) := I(z,M) + (n-z)\exp(-N p_f)$$

which is an upper bound of $J_M(X)$. We already remarked that for fixed $M$ only few $z$ need to be checked if $I(z,M) > Q$ 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 interval 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 $p_r(X,i)$ and $p_r(X,1_j)$ for all $f \in F$)

Begin
Y0 := (X,0);  
F_0 := F;  
ANALYSIS (Y0,F_0);  
Y1 := (X,1);  
F_1 := F;  
ANALYSIS (Y1,F_1);  
F_0_1 := Ordered list with the corresponding 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);  
While (N_old - N_new) > $\sigma$ do
  (\(\sigma\) is user defined)
  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);  
        $x_i := y$;  
        ANALYSIS (X,F);  
        SORT(F);  
        NORMALIZE(N_new,nf);  
      END;
  end;
end;

In the next section we discuss the performance and some applications.

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.

<table>
<thead>
<tr>
<th>Circuit</th>
<th>CPU time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>* S1</td>
<td>300</td>
</tr>
<tr>
<td>* S2</td>
<td>600</td>
</tr>
<tr>
<td>* C2670</td>
<td>1,200</td>
</tr>
<tr>
<td>* C7552</td>
<td>2,000</td>
</tr>
</tbody>
</table>

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.

![Fault coverage vs. pattern count (S1)](image)

**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 successful 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:**

- Agrawal, V.D.; Agrawal, P.: On Improving the Efficiency of Monte Carlo Test Generation; in: FTCS 5, 1975
- Bennett, R.G.: Design of Testable Logic Circuits, Addison-Wesley, 1984
- Savir, J.; Random Pattern Testability; in: IEEE, Trans. on Comp., Vol. C-33, No. 1, 1984
- Brzozowski, F. et al.: Accelerated ATPG and fault grading via testability analysis; in: Proc. ISCAS 85, Kyoto 1985