The Random Pattern Testability of Programmable Logic Arrays

Hans-Joachim Wunderlich

Universität Karlsruhe
Institute for Computer Design and Fault Tolerance
(Prof. Dr. D. Schmid)
7500 Karlsruhe, FRG
Phone: (0721) 608-4257

Abstract: An efficient Monte Carlo Algorithm is presented estimating the detection probability of each stuck-at fault of a PLA. Furthermore for each primary input of the PLA the optimal probability is computed to set this input to logical "1". Using those unequiprobable input probabilities the necessary test set can be reduced by orders of magnitude. Thus a self test by optimized random patterns is possible even if the circuit contains large PLAs preventing a conventional random test.

Keywords: PLAs, random test, fault detection probabilities, self test

1. Introduction

Many design techniques have been proposed in order to enhance the testability of PLAs. In [FuKi81] and in [DaMu81] an additional row and column are proposed in order to use universal test sets, and some textbooks are discussing testable PLAs as a subject of its own [Fuji85]. There are also some attempts to build expert systems selecting an appropriate test strategy [BrZh85].

A common test strategy for semicustom circuits is self testing based on 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 ([Tsa183], [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.

But often semicustom circuits contain PLAs, which resist a random test due to low fault detection probabilities. Especially PLAs with large product terms will need an exorbitant size of a random test set. In [LIGH86] a design method based on simulated annealing is proposed minimizing the size of the product terms and enhancing the testability of the PLA. But due to the intended function of the PLA this approach will not always yield satisfying results. Therefore methods are required to estimate fault detection probabilities and to minimize random test lengths as well. In this paper solutions of both problems are proposed.

Currently significant work is done to estimate fault detection probabilities in circuits with random logic. In PREDICT [SETH86] and PROTEST [Wu85] an analytical way is used, whereas STAFAN [AgJa84] estimates detection probabilities by value counting. But seeing that the underlying fault detection problem is NP-hard [IbSa75] both methods suffer from a large trade-off between efficiency and accuracy.

Karp and Luby found a fully polynomial Monte Carlo algorithm estimating the number of minterms fulfilling a boolean function in disjunctive form [KaLu83]. Lieberherr [Lieb84] conjectured that based on this algorithm computing detection probabilities for circuits in random logic can be solved polynomially. However due to the already mentioned NP-completeness this is not true.

But the simple structure of a PLA can be used to represent fault detection probabilities as the difference of the probabilities of the truth of two boolean functions in disjunctive form. Those functions can be stated in linear effort, and the probabilities can be computed in polynomial time by the Karp-Luby algorithm. This procedure is presented in the next section, and in section 3 some details of an efficient implementation are discussed.

It is known that fault detection probabilities can increase and the necessary number of test patterns can decrease, if each primary input is set to logical "1" with its specific optimal input probability ([LBGG86], [Wu86]), in [Wu87] a procedure is presented to compute those probabilities. In section 4 it is shown that the Monte-Carlo algorithm can be modified in order to compute such optimal input probabilities without significant overhead. In section 5 some applications and results are discussed.

2. Estimating fault detection probabilities by a Monte-Carlo algorithm

The Karp-Luby algorithm estimates the number of input combinations for which a boolean function, presented in disjunctive normal form, assumes the value *true*. It is fully polynomially, i. e. for every input triple (ε,δ,w), where w is the description of the function and f(w) is the required number, and where $\varepsilon > 0$ and $0 < \delta < 1$, the algorithm produces as output a number

$$\tilde{f}_{\varepsilon,\delta}(w)$$

such that

$$\Pr\left[\frac{|\tilde{f}_{\varepsilon,\delta}(w) - f(w)|}{f(w)} > \varepsilon\right] < \delta$$

and the execution time of the algorithm is bounded by a polynomial in $1/\varepsilon$, $1/\delta$ and the length of w.

Now we first describe how the function of a PLA usually is represented in logic design and minimization by the cubical calculus [Roth80], then we use this notation in order to explain the Karp-Luby algorithm, and at the end of this section fault detection probabilities are discussed.

2.1 The representation of PLAs by the cubical calculus

For the sake of simplicity we restrict our attention to single output PLAs, for multiple outputs the generalizations are straightforward.

Let F be a boolean function with n input variables $e_1,\dots,e_n$. For each product term P_i $(i=1,\dots,m)$ of F a cube $C_i := (c_1,\dots,c_n)$ is defined. The components c_i are as follows:

$$c_i := \begin{array}{l} 0, \text{ if } e_i \text{ is negated in } P_i \\ 1, \text{ if } e_i \text{ is positive in } P_i \\ 2, \text{ if } e_i \text{ is not part of } P_i \end{array}$$

The set of cubes $C := \{C_i \mid i=1,\dots,m\}$ is called the cover of F, and it describes the function uniqely. A minterm is a cube, where all components have either the value "0" or the value "1". The cube C contains the cube D, $C \supset D$, if $c_j \neq 2 \Rightarrow c_j = d_j$, holds for all $j=1,\dots,n$, .

Using this relation we define $|C|$ as the number of minterms contained in C, and $\cup C$ is the set of all minterms of F.

2.2 The Monte-Carlo algorithm

With the already mentioned parameters ε and δ we now use the Karp-Luby algorithm to estimate the probability P(*C*) that a boolean function with cover *C* is true:

```
draws := 0; trials := 0;
repeat until  $draws \geq \max(\frac{500m}{\delta}, \frac{5m}{\varepsilon^2\delta})$
      begin
      chose $i \in \{1,..,m\}$ with probability $\frac{|C_i|}{\sum_{j=1}^{m}|C_j|}$
      chose a random minterm S of $C_i$;
      k := 0,
      trials := trials+1;
      repeat until a j is chosen such that S is
      in $C_j$
            begin
            draws := draws+1;
            k := k+1;
            chose j at random in {1,...,m}
            end
      $Y_{trials}$ := k;
      end
```

$$X := \frac{\sum_{i=1}^{m}|C_i|}{m} * \frac{\sum_{j=1}^{trials} Y_j}{trials}$$

With this algorithm X is an estimation for P(*C*). In [KaLu83] it is shown that this is a fully polynomial algorithm in ε, δ and m*n, for fixed ε, δ running in O(m*n) time.

2.3 The detection probability of stuck-at faults

For each primary input e_j (j=1,..,n) the following covers are defined:

$$C_j^0 := \{C \in \mathcal{C} \mid c_j = 0\}$$

$$C_j^1 := \{C \in \mathcal{C} \mid c_j = 1\}$$

$$C_j^2 := \{C \in \mathcal{C} \mid c_j = 2\}$$

We define the cube $C^{0,j} := (c_1,..,c_{j-1},0,c_{j+1},...,c_n)$ and the cubes $C^{1,j}$ and $C^{2,j}$ in an analogous way. Finally let be

$$C_j^{0,2} := \{C^{2,j} \mid C \in C_j^0\} \text{ and}$$
$$C_j^{1,2} := \{C^{2,j} \mid C \in C_j^1\}.$$

First we are discussing stuck-at faults at the primary inputs of the PLA.

Case I: st-0 at the j-th input.

a) If the j-th input is nowhere negated then C_j^0 is empty and this is a diminishing fault. The detection probability P_f is computed by

$$P_f = P(C) - P(C_j^2)$$

b) If the j-th input only occurs negated at the product terms, then C_j^1 is empty and this is an enlarging fault. P_f is computed by

$$P_f = P(C_j^{0,2} \cup C_j^2) - P(C)$$

c) If the j-th input occurs both positive and negative then there may be some wrong minterms added, and some others may be taken away. The probability that some minterms are added is

$$P_{fad} = P(C_j^1 \cup C_j^2 \cup C_j^{0,2}) - P(C)$$

A little bit more complicated is the probability that some minterms are missing, since those minterms can be added by $C_j^{0,2}$:

$$P_{fsu} = P(C_j^1 \cup C_j^2 \cup C_j^{0,2}) - P(C_j^2 \cup C_j^{0,2})$$

And now we have $P_f = P_{fad} + P_{fsu}$.

In an analogous way one can compute the next case:

Case II: st-1 at the j-th input.

a) C_j^0 is empty: enlarging fault,

$$P_f = P(C_j^{1,2} \cup C_j^2) - P(C)$$

b) C_j^1 is empty: diminishing fault,

$$P_f = P(C) - P(C_j^2)$$

c) miscellaneous:

$$P_{fad} = P(C_j^1 \cup C_j^2 \cup C_j^{1,2}) - P(C)$$

$$P_{fsu} = P(C_j^1 \cup C_j^2 \cup C_j^{1,2}) - P(C_j^2 \cup C_j^{1,2})$$

$P_f = P_{fad} + P_{fsu}$.

Finally we have to deal with the faults at the product terms, i. e. the cubes:

Case III: stuck-at faults of variable e_j at C_i:

a) $e_j = 0$ and st-0 or $e_j = 1$ and st-1

$$P_f = P(C \setminus \{C_i\} \cup \{C_i^{2,j}\}) - P(C)$$

b) $e_j = 0$ and st-1 or $e_j = 1$ and st-0

$$P_f = P(C) - P(C \setminus \{C_i\})$$

Up to this point we can now compute the fault detection probabilities by the Monte-Carlo algorithm for covers *C*. But since for different faults the needed covers use the same cubes to a large extent something more can be done to enhance efficiency.

3. Implementation of the estimation algorithm

Let Ft be the set of all stuck-at faults under interest. For each fault f ∈ Ft one has to deal with 2 covers (case I a, b; case II a, b; case III) or with 4 covers (case I c, case II c). For different faults those covers are not necessarily different too, e. g. the value of P(*C*) is needed very often. Of course one cover is only treated once.

Now let $A := \{C_i \mid i=1,..,t\}$ be the set of all covers the probabilities of which have to be computed. Those t different covers again contain the same cubes to a large extent. Therefore we set

$$H := \cup A = \{C \mid \exists h \in \{1..t\}\ C \in C_h\} = \{H_i \mid i=1,..,u\},$$

which is the collection of all involved cubes. Now the Karp-Luby algorithm can be parallelized widely:

```
draws(1..t) := 0; trials(1..t) := 0;
repeat until $A = \emptyset$
      begin
      $H := \cup A$;
      chose $i \in \{1..u\}$ with probability $\frac{|H_i|}{\sum_{j=1}^{u}|H_j|}$
      chose a random minterm S in $H_i$;
      Set $A' := \{C \in A \mid H_i \in C\}$;
      For all $C_k \in A'$ do trials(k) := trials(k) +1;
```

```
repeat until $A' = \emptyset$
        begin
        d := 0;
        $H' := \cup A'$;
        repeat until a $H \in H'$ is chosen such
                        that S is in H
                begin
                d := d+1;
                chose $H \in H'$ at random
                end
        For all k with $H \in C_k$ do
                begin
                $d' := d^* |C_k| / |H'|$
                $Y_{trials(k)}(k) := d'$;
                draws(k) := draws(k) + d'
                $A' := A' \setminus \{C_k\}$
                If draws(k) ≥
                   $\max(500^* |C_k| / \delta, 5^* |C_k| / \varepsilon^2 \delta)$
                then begin
                        $P(C_k) := \frac{\sum_{C \in C_k} |C|}{|C_k|} \cdot \frac{\sum_{j=1}^{trials(k)} Y_j(k)}{trials(k)}$
                        $A := A \setminus \{C_k\}$
                        end
                end
        end
end.
```

4. Optimal input probabilities

Up to this point we have constructed an efficient procedure to estimate fault detection probabilities for a set of faults. The algorithm is presented under the assumption that each primary input e_i ($i=1,..,n$) is set to logical "1" with probability 0.5. Now we want to stimulate the inputs with unequiprobable patterns according to $<x_1,..,x_n>$, $x_i \in [0,1]$ in order to enhance testability. In this case only slight modifications of the presented algorithms are necessary, the details are left to the reader.

In this section we summarize the solution of the optimizing problem in [Wu87], which results in a drastical reduction of the size of random test sets, and discuss its application to PLAs. Let δ be the confidence of a random test of length N, that is the probability to detect all faults $f \in Ft$ 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

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

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 this formula will only provide an upper bound.

For each fault $f \in Ft$ the detection probability $p_f(X)$ depends on the tupel of input probabilities $X := <x_i \mid i \in I>$. Therefore the formula above turns into

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

This formula expresses the probability that all faults are detected by N patterns with the distributions of X.

Using some well known approximations this is transformed into

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

This describes the objective function, and we call a tupel X of input probabilities ***optimal with respect to N***, if

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

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

In the {0,1}-space expectation value and probability coincide, and the stochastical optimizing problem reduces to a deterministical 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. 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). 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. In [Wu87] it is shown that the objective function is strictly convex with respect to one single variable. Hence for each fixed $x_1,...,x_{i-1}, x_{i+1},...,x_n$ there exists exactly one $x_i \in [0,1]$ with minimum $J_N(x_1,...,x_i,...,x_n)$.

Notation:

Let $Z := <z_1,\dots,z_n>$, and $y \in [0,1]$. We write $f(Z,y_{|i}) := f(z_1,\dots,z_{i-1},y,z_{i+1},\dots,z_n)$.

One easily verifies that

$$p_f(X) = p_f(X, 0_{|i}) + x_i(p_f(X,1_{|i})-p_f(X,0_{|i})).$$

Hence

$$\frac{dJ_N(X,y_{|i})}{dy} = \sum_{f \in Ft} -N(p_f(X,1_{|i})-p_f(X,0_{|i}))e^{-Np_f(X,y_{|i})}$$

and

$$\frac{d^2J_N(X,y_{|i})}{d^2y} = \sum_{f \in Ft} N^2(p_f(X,1_{|i}) - p_f(X,0_{|i}))^2 e^{-Np_f(X,y_{|i})} > 0.$$

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 now we have:

For each $\overline{X} \in [0,1]^I$ 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.

Since all derivations of the objective function are explicitly available this minimum point can be computed by simple iterations like the Newton algorithm or the regula falsi.

And now it turns out that we need only to compute the values $p_f(X,1_{|j})$ and $p_f(X,0_{|j})$ for all faults additionally. This is done by the same Monte-Carlo algorithm as before restricted to the two smaller covers

$$C_j^{1,2} \cup C_j^2 \text{ and } C_j^{0,2} \cup C_j^2$$

instead of the cover C. Thus the effort to compute the optimal probability for a primary input is in general only a little higher than the effort needed to estimate fault detection probabilities $p_f(X)=p_f(X,0_{|j})+x_i(p_f(X,1_{|j})- p_f(X,0_{|j}))$

The complete optimizing procedure successively tries to optimize each primary input of the PLA, and starts again if one cycle has provided a significant enhancement of testability.

5. Applications and results

The estimation algorithm is implemented in [Kara87], and the optimization procedure is part of the tool PROTEST (Probabilistic testability analysis). The optimization yields in a reduction of test sets by several orders of magnitude (see [Wu87] e.g.). As an example see the PLA of Fig. 1:

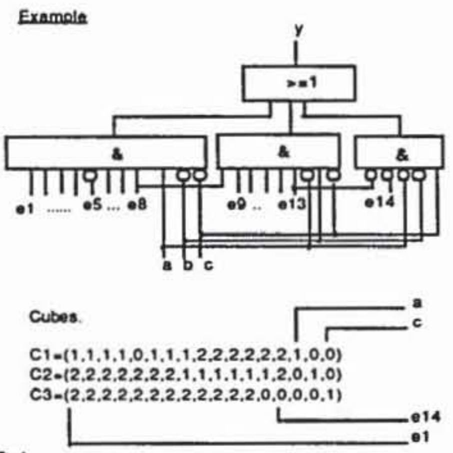
Example
y
>=1
&
&
&
e1 ...... e5 ... e8
e9 .. e13
e14
a b c
Cubes.
a
c
C1=(1,1,1,1,0,1,1,1,2,2,2,2,2,2,1,0,0)
C2=(2,2,2,2,2,2,2,1,1,1,1,1,1,2,0,1,0)
C3=(2,2,2,2,2,2,2,2,2,2,2,2,0,0,0,0,1)
e14
e1

Fig. 1: PLA

For a conventional random Test with confidence δ = 0.98 the presented algorithm would require 14 664 equiprobable random patterns. Using the optimizing algorithm based on Newton iteration after 6 cycles only 750 optimized random patterns are necessary. The computed input probabilities are:

e1 : 0.823	e8 : 0.825	a : 0.480
e2 : 0.825	e9 : 0.797	b : 0.422
e3 : 0.827	e10 : 0.799	c : 0.218
e4 : 0.829	e11 : 0.800	
e5 : 0.170	e12 : 0.802	
e6 : 0.832	e13 : 0.680	
e7 : 0.833	e14 : 0.445	

Table 1: Optimized input probabilities

These predictions were validated by fault simulation. By a set of 750 optimized random patterns a complete fault coverage (100 %) was achieved, whereas conventional sets of 750 patterns only lead to 70 % thru 75 %.

Self test by random patterns is the main goal of the optimizing approach. A self test modul for unequiprobable random patterns similar to the well known BILBO is presented in (Wu86) and (Wu87a).

Besides this direct test application the estimation procedure is also used to support logic minimization [RoWu87]. Based on the Monte-Carlo algorithm partitioning variables are selected in order to reduce the PLA to parts which can be handled by a logic minimizer. Here it turns out that test and minimization requirements are not contradictory.

6. Conclusions

A Monte Carlo algorithm was presented estimating the random pattern testability of a PLA efficiently. Moreover based on this algorithm for each primary input a specific optimal probability can be computed to set it logical "1". Using those input probabilities the necessary size of a random test set can decrease significantly.

Literature

AgJa84 Jain, S. K.; Agrawal, V.D.: STAFAN: An alternative to fault simulation; in: Proc. 21st Design Automation Conference, 1984

BrZh85 Breuer, M.A.; Zhu, X.: A Knowledge Based System for Selecting a Test Methodology for a PLA; in: Proc. 22nd Design Automation Conference, Las Vegas, 1985

DaMu81 Daehn, W.; Mucha, J.: A hardware approach to self-testing of large programmable logic arrays; IEEE Trans. Comp., Vol. C-30, 1981, pp. 829-833

Fuji85 Fujiwara, H: Logic Testing and Design for Testability; The MIT Press, 1985

FuKi81 Fujiwara, H; Kinoshita, K.: A design of programmable logic arrays with universal tests; IEEE Trans. Comp., Vol. C-30, 1981 pp. 823-828

IbSa75 Ibarra, O.H.; Sahni, S.K.: Polynomially Complete Fault Detection Problems; IEEE Trans. Comp. Vol. C-24, No. 3, March 1975

Kara87 Karamarkos, N.: Die Zufallstestbarkeit von PLA's als Partitionierungskriterium; Diploma Thesis, Fakultät für Informatik, Universität Karlsruhe, 1987

KaLu83 Karp, R.M., Luby, M.: Monte-carlo Algorithms for Enumeration and Reliability Problems; Proc. Annual Symp. on Foundations of Computer Science, 1983, pp. 56-64

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

LIGH86 Lighthard, M. M. et al.: Design-for-Testability of PLAs Using Statistical Cooling; Proc. 23rd Design Automation Conference, 1986

LBGG86 Lisanke, R. et al.: Testability-Driven Random Pattern Generation; in: Proc. ICCAD, November 1986

Roth80 Roth, P.: Computer Logic, Testing, and Verification; Pitman, 1980

RoWu87 Rosenstiel, W.; Wunderlich, H.-J.: PLA partitioning under testability and minimization constraints; in preparation

SETH86 Seth, S. C. et al.: An Exact Analysis for Efficient Computation of Random-Pattern Testability in Combinational Circuits; in: FTCS 16, 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

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.: On Computing Optimized Input Probabilities for Random Tests, Proc. 24th Design Automation Conference, Miami, 1987

Wu87a Wunderlich, H.-J.: Self test using unequiprobable random patterns;FTCS-17, Pittsburgh 1987

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