A new Heuristic Algorithm for Estimating Signal and Detection Probabilities

Musaed A. Al-Kharji Sami A. Al-Arian
Computer Science & Engineering Department
University of South Florida
Tampa, Fl. 33620

Abstract

The problem of computing signal probabilities of digital circuits arises in the context of random testing, pseudorandom testing, and testability analysis. However, it has been known that computing signal probabilities is #P-complete [3]. This implies that the problem might be intractable even if $P = NP$. Thus, any practical method will estimate such probabilities instead of computing the exact values.

This paper presents a simple but effective algorithm for estimating signal probabilities. It provides significantly better estimates of signal probabilities than the weighted averaging algorithm and most importantly is linear in the product of circuit size and the number of primary inputs. Based on this algorithm, the detection probabilities of stuck-at faults are estimated.

Experimental results using ISCAS benchmark circuits show the effectiveness and the improvement of this technique over the simple algorithm as well as the weight averaging algorithm. The correlation coefficients of the results are extremely good and the algorithm is very fast.

I. Introduction

With the advent of VLSI technology, test pattern generation (TG) for high fault coverage is regarded as one of the most difficult problems in the field of VLSI testing. It has been known for sometime that the process of TG is NP-complete. TG can consume CPU time and memory at a rate which increases at least as the square of the number of gates in the circuit [6]. For VLSI circuits, this is a serious limitation. This increases the feasibility of design for testability in reference to built-in self test (BIST). Random/pseudorandom self testing is a test strategy used in BIST. However, this strategy introduces the problem of test quality verification especially when high fault coverages are required. The computation of test quality involves the identification of hard to detect faults, namely, those faults whose detection probability falls below a given threshold. The detection probability of a given fault is the probability that a randomly chosen input vector detects the fault. The computation of detection probabilities of a given fault involves the computation of signal probabilities in the network. The signal probability of a line l in a network is the probability of line l having a value 1 on a randomly selected vector.

Actually, the computation of the detection probabilities of a fault could be reduced to the computation of signal probability of an auxiliary gate, whose output is 1 if the sensitizing conditions of that fault are satisfied [7]. Consequently, the problem of computing the signal probabilities is of central importance in random pattern testability analysis.

It has long been established that computing signal probabilities is #P-complete [3]. This implies that the problem might be intractable even if P=NP. Current algorithms for signal probability can either compute the exact value or estimate such probabilities. The exact computation algorithms suffer from undue complexity, while estimating algorithms suffer from imprecision.

Parker and McCluskey [1,2] presented the foundations of signal and detection probabilities. Their algorithms for computing the exact signal probabilities use the exponent suppression method. This method is not useful because it very often requires exponential space.

PREDICT [8] calculates signal probabilities without simulation by using conditional probabilities. A graph approach is used to exactly compute these probabilities using Shannon's expansion. An attempt was made to modify the algorithm used in PREDICT so that the exact values of the detection probability of single stuck-at faults could be computed [10]. However, the computational complexity of these approaches increases exponentially with cardinality of the number of reconvergent fanout nets at any node. Also, the enumeration algorithms presented in [12] for computing signal and detection probabilities combine the notion of PREDICT and have exponential time complexity.

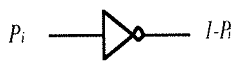
1066-1395/97 $10.00 © 1997 IEEE

The Monte Carlo algorithms [4], on the other hand, estimate such signal probabilities. However, the expected accuracy of the estimates depends on the number of experiments performed. In general, the number of experiments needed to obtain reasonable estimates would grow at least as fast as a polynomial of a large degree equal to the number of inputs [11].

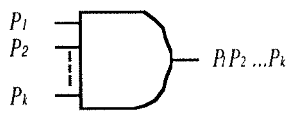
The cutting algorithm presented in [7] provides lower and upper bounds on the signal probabilities. Its objective is to turn the combinational network into a tree, by cutting reconvergent fanout branches, and inserting equivalent bounds at the cut points, which will guarantee that all the signal probability bounds computed on this tree will enclose the true values. The advantage of doing this is the reduction of the computational complexity (time and space); while the disadvantage is that the output is just a bound and its effectiveness depends on the cutting choices. There are two cutting algorithms given in [7], the full-range and the restricted-range algorithms. The computation complexities of the restricted-range for signal probabilities is $O(N^3)$.

The simple algorithm, used in COP [5] and extended in [11], estimates the signal probabilities using signals' independence at the node inputs. This algorithm runs in linear time, and is easy to implement. The following is a formal description of the algorithm:

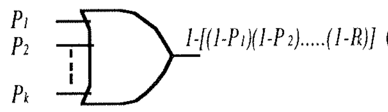
The Simple Algorithm
1. Assign signal probabilities of 1/2 to all the primary inputs.
2. Proceed from the inputs to the outputs and compute the signal probability of each output gate using the following formulas:



(1)



(2)



(3)

In fact, if a network has no reconvergent fanout then the simple algorithm will compute the exact values of signal probabilities. Otherwise, it will suffer from imprecision.

The weighted averaging algorithm [11] is an extension of the simple algorithm and provides a well received overall root mean square (RMS) deviation from the exact values of signal probabilities. The time complexity of the algorithm is linear in the product of the size of the circuit and the number of primary inputs.

This paper presents a new and effective algorithm based on a new set of inference rules for estimating signal probabilities, that we shall call the possibilistic algorithm. The proposed algorithm is also an extension of the simple algorithm. However, It provides significantly better estimates of signal probabilities than the weighted averaging algorithm and it is also linear in the product of circuit size and the number of primary inputs. Based on this algorithm, the detection probabilities of stuck-at faults are estimated.

Experimental results offer empirical evidence indicating the quality of results produced by these algorithms on a commonly accepted set of ISCAS benchmark circuits.

II. The Possibilistic Algorithm for Estimating Signal Probabilities

The major difficulty in computing the signal probabilities is reconvergent fan-outs. In fact, if a circuit has no reconvergent fan-outs, then the independent formulas (1), (2), and (3) used by the simple algorithm will compute the exact signal probabilities in a linear time. However, if a node fans out and reconverges at a gate *G*, then the signal probability computation of the simple algorithm for all nodes driven by gate *G* is more likely to be deviated from the exact values. The idea of our technique lies in the observation that the deviated computation of the simple algorithm for any gate *G* could be caused by: a) a reconvergence at gate *G*, b) a reconvergence at a gate in its cone of influence, or c) a combination of them. The possibilistic algorithm distinguishes between these causes and uses a proper inference rule to reduce the error in every case.

The proposed technique for estimating signal probabilities for every output gate is given in the following steps. (For the sake of clarity, the steps will be applied, below, to the circuit in Figure 1.)

Possibilistic Algorithm:
Input: A combinational network with n primary inputs, $I_1, I_2, ... ,I_n$.
Output: An estimate of the signal probabilities of each output nodes.
Procedure:
Step 1: Run the simple algorithm (S_Alg) to estimate the signal probability (SP) at the output node of each gate. These estimates are denoted by *S* in fig. 1.
Step 2: For each primary input I_i, $1 \leq i \leq n$, perform the following:

a) Set the input I_i to 0 and run the S_Alg, let us denote the resulting SP at the output gate by S_Alg(I_i=0).

b) Set the input I_i to 1 and run the S_Alg once again, yielding the SP S_Alg(I_i=1). In a) and b), when we set input I_i to 0 or 1, all other inputs I_j, $1 \le j \le n$, $j \ne i$, are set to $\frac{1}{2}$.

c) Compute the average of S_Alg(I_i=0) and S_Alg(I_i=1), denote the result as p_i.

$$p_i = \frac{S\_A\lg(I_i = 0) + S\_A\lg(I_i = 1)}{2} \qquad (4)$$

Step 3: Calculate the level of every gate in the network and sort the gates in non-decreasing order according to their levels.

Step 4: Proceeding from the inputs to the outputs of each gate, perform the following steps starting from the gates with the lowest level:

a) Compute the expected-tuple (E-tuple) of each output gate using the P-tuples of its inputs and the independent formulas (1), (2), and (3).

b) For every component p_i, $1 \le i \le n$, in P-tuple do:Compare it with S, computed in *Step 1.* If they are equal, mark it with 0. If they are not, then compare it with its expected value e_i in the E-tuple and mark it with 1 if they are equal, else mark it with 2. By doing this, we will have a Mark-tuple (M-tuple) with components m_i, $1 \le i \le n$.

c) Count the number of 1s and 2s in the M-tuple, let us denote these counts by ctr1 and ctr2, respectively.

d) Compute the no-dependent (ND) value by using independent formulas and the estimated signal probabilities of its inputs from previous level.

e) Apply the inference rules listed in Figure 2. (These rules were developed and refined through empirical and experimental results).

By performing *Step 2* for each primary input, we will have a probability tuple (P-tuple) of n components for each output gate. Each component p_i is the resulting computation of the average setting of expression (4) with respect to each primary input. The component p_i of gate *G* will avoid errors due to multiple dependencies on primary input I_i.

Table-I lists the P-tuples for the circuit in fig. 1. The components are listed in order, i.e. $(p_1, p_2, p_3, \ldots\ldots, p_n)$ for a circuit with *n* primary inputs.

In Step 4, we compute two more tuples, namely, the Expected-tuple (E-tuple) and the Mark-tuple (M-tuple). For example, the P-tuples of the inputs of gate U2, I_1 and *U1*, are (1/2, 1/2, 1/2) and (3/4, 3/4, 3/4), respectively.

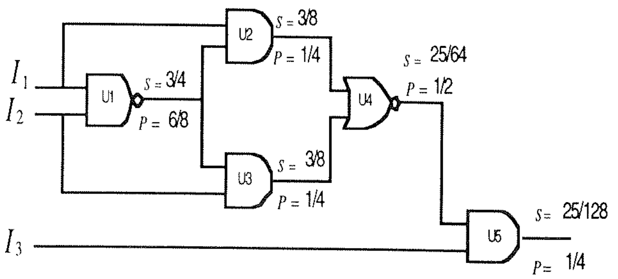


Fig. 1 (Network example). S denotes the estimate computations of the Simple algorithm while P denote the output estimates of the possibilistic algorithm.

The E-tuple of the output gate, U2, is $(\frac{1}{2}*\frac{3}{4}, \frac{1}{2}*\frac{3}{4}, \frac{1}{2}*\frac{3}{4})$ = (3/8, 3/8, 3/8). The M-tuple records useful information about the type of dependence, with respect to every primary input, at a current gate under processing G. If a primary input I_i does not influence gate G, then S = p_i and will be marked with 0 in the M-tuple. However, if I_i influences gate G then $S \ne p_i$. In this case, we have three possibilities for this deviation:

- The input I_i does not reconverge at gate G, but it reconverges at a gate(s) in the cone of influence of G and its effect causes $S \ne p_i$. We mark this case by 1 in M-tuple.
- The input I_i reconverges at gate G but not at any node in its cone of influence. This case is marked by 2.
- A combination of the above two cases. Here, we also mark it with 2. Table-I list the M-tuples for the circuit in fig. 1.

The calculation of no-dependent (ND) proceeds from primary inputs (level 0) to each output of the gate in a recursive fashion, by applying the independent formulas (1),(2), and (3) to the estimates of its inputs. For example, for gate U1, ND = $1 - \frac{1}{2}*\frac{1}{2} = \frac{3}{4}$. However, ND of gate U2 depends on the signal probability estimation for gate U1. Assuming that, this algorithm estimates a signal probability equal to $\frac{3}{4}$ for gate U1, then ND of gate U2 is $\frac{1}{2}*\frac{3}{4} = \frac{3}{8}$. Similarly the NDs for other gates of Fig.1 could be calculated in the same way.

The results of applying the inference rules of Fig. 2 are shown in Table II. For example, for gate U1, ctr1 = 0 and ctr2 = 0, which satisfy rule-1(*Case 1*). So, P = S = $\frac{3}{4}$, where P denotes the estimate of the possibilistic algorithm. For gate U2, ctr1 = 0 and ctr2 = 1, which

TABLE I
Probabiliy, Expect, and Mark tuples for Fig. 1

Nodes	Probability-Tuples	Expect-Tuples	Mark-Tuples
I_1	$(\frac{1}{2},\frac{1}{2},\frac{1}{2})$	$(\frac{1}{2},\frac{1}{2},\frac{1}{2})$	(0, 0, 0)
I_2	$(\frac{1}{2},\frac{1}{2},\frac{1}{2})$	$(\frac{1}{2},\frac{1}{2},\frac{1}{2})$	(0, 0, 0)
I_3	$(\frac{1}{2},\frac{1}{2},\frac{1}{2})$	$(\frac{1}{2},\frac{1}{2},\frac{1}{2})$	(0, 0, 0)
U1	$(\frac{3}{4},\frac{3}{4},\frac{3}{4})$	$(\frac{3}{4},\frac{3}{4},\frac{3}{4})$	(0, 0, 0)
U2	$(\frac{1}{4},\frac{3}{8},\frac{3}{8})$	$(\frac{3}{8},\frac{3}{8},\frac{3}{8})$	(2, 0, 0)
U3	$(\frac{3}{8},\frac{1}{4},\frac{3}{8})$	$(\frac{3}{8},\frac{3}{8},\frac{3}{8})$	(0, 2, 0)
U4	$(\frac{7}{16},\frac{7}{16},\frac{13}{32})$	$(\frac{15}{32},\frac{15}{32},\frac{25}{64})$	(2, 2, 0)
U5	$(\frac{7}{32},\frac{7}{32},\frac{25}{128})$	$(\frac{7}{32},\frac{7}{32},\frac{25}{128})$	(1, 1, 0)

satisfy rule-3, thus, $P = \frac{1}{4}$.

In rule-3, the variable $effect2 = \sum_{mi=2} (pi - ei)$, represents the total deviation of the P-tuple from the E-tuple of those components which are marked with 2. The function sign( ), used in rule-3, return the sign of its argument.

```
Case 1: if (ctr1 = 0 and ctr2 = 0) then { P = S }

Case 2: if (ctr1 ≠ 0 and ctr2 = 0) then {P = ND}

Case 3: if (ctr1 = 0 and ctr2 ≠ 0) then {
            if (ctr2 = 1)then P = $p_i$
            else { effect2 = $\sum_{mi=2} (p_i - e_i)$
                   P = ND + effect2 + sign(effect2)*
                       (effect2/ctr2)
                 }
                                        }
Case 4 : if (ctr1 ≠ 0 and ctr2 ≠ 0) then {
                   effect2 = $\sum_{mi=2} (p_i - e_i)$
                   P = ND + effect2 + sign(effect2)*
                       (S - ND)* $\frac{effect2}{S}$
                                        }
```

Figure 2. Inference rules for estimating signal probabilities

TABLE II
Signal Probabilities for Fig. 1

Nodes	Simple -Alg.	Weighted -Alg.	Possibilistic -Alg.	Exact
I_1	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
I_2	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
I_3	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
U1	$\frac{3}{4}$	$\frac{3}{4}$	$\frac{3}{4}$	$\frac{3}{4}$
U2	$\frac{3}{8}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$
U3	$\frac{3}{8}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$
U4	$\frac{25}{64}$	$\frac{7}{16}$	$\frac{1}{2}$	$\frac{1}{2}$
U5	$\frac{25}{128}$	$\frac{7}{32}$	$\frac{1}{4}$	$\frac{1}{4}$

By checking the results in Table II, one will note that the estimates of this algorithm, for the network of fig.1,are equal to the exact values. Also, we ran this algorithm on several benchmark networks given in [18], and it provides a well received overall root mean square deviation from the exact values. Table III gives a summary of the results and shows the improvement of this algorithm over both the simple as well as the weighted averaging algorithm.

III. Estimating Sensitization and Detection Probabilities

Similar to the signal probability algorithm, the sensitization procedure first assigns a sensitization probability of 1.0 to all primary outputs since they are completely observable. Having made the output assignments, the procedure then repeatedly traces back the sensitization probabilities along lines with the same connection and calculates a gate's input sensitization probabilities until all sensitization probabilities have been established.

Unlike signal probabilities, the sensitization probabilities of fanout branches with a common stem are not always equal. However, for a node which is not a fanout stem we can assume that it inherits the sensitization probability of its unique connection. For a fanout stem, we approximate its sensitization by:

$$S(stem) = S_{x1} \oplus S_{x2} \oplus \ldots \oplus S_{xn} \quad (5)$$

where the associative operation $\oplus$ is defined as:

$$a \oplus b = a + b - a * b \quad (6)$$

TABLE III
A COMPARISON BETWEEN SIMPLE, WEIGHTED, AND POSSIBILISTIC ALGORITHMS

CIRCUIT	Exact_Computation E = Exhaustive M = Monte Carlo	RMS of Deviations Simple_ Algorithm	Weighted _ Algorithm	Possibilistic_ Algorithm	% Improvement Weighted over Simple	%Improvement Possibilistic over Simple	Correlation Coefficient from Possibilistic
4-Bit ALU	E	0.0672	0.0563	0.0182	19.36	269.23	0.9992
4-Bit Compartor	E	0.085	0.0308	0.0237	175.97	258.65	0.997
9-Bit Parity	E	0.0675	0.0549	0.03	22.95	125	0.9957
C432	M	0.0941	0.0853	0.0551	10.31	70.78	0.9788
C499	M	0.0502	0.0426	0.0394	17.84	27.41	0.9761
C880	M	0.0360	0.0351	0.0322	2.56	11.80	0.993
C1355	M	0.0835	0.0719	0.0652	16.13	28.07	0.9671

S_{x1} to S_{xn} represent the sensitization probabilities of fanout branches driven by the common stem.The sensitization probability algorithm proceeds from primary outputs to primary inputs in a recursive fashion. The formula for estimating the gate input sensitization probabilities is:

$$Se_i = S_{\text{gate output}} * [f(Pe_1, Pe_2, \dots, Pe_{i-1}, 0, Pe_{i+1}, \dots, Pe_{n-1}, Pe_n) \oplus f(Pe_1, Pe_2, \dots, Pe_{i-1}, 1, Pe_{i+1}, \dots, Pe_{n-1}, Pe_n)] \quad (7)$$

where Se_i is the gate input sensitization probability to be determined, Pe_1 to Pe_n are the gate input signal probabilities, f is the Boolean function applied to the signal probabilities according to the gate type, and the $\oplus$ signifies the associative operation as previously defined. The above-mentioned associative operation was defined differently in [9]. The proposed algorithm was tested with both formulas and the above-mentioned definition - which describes the probabilistic disjunction of two signals - consistently resulted in higher correlation coefficients and smaller absolute average differences between the estimated and the exact fault detection probabilities.

Using formula (7), the specific gate input sensitization probability formulas for various kinds of gates could be calculated easily. For example, for an AND gate, the function f is the product of the gate input signal probabilities, and the first part of the associative operation reduces to zero, simplifying the equation to:

$$\begin{aligned} Se_i(AND) &= S_{\text{gate output}}(AND) * [f(Pe_1 * Pe_2 * \dots * Pe_{i-1} * 0 * Pe_{i+1} * \dots * Pe_{n-1} * Pe_n) \oplus f(Pe_1 * Pe_2 * \dots * Pe_{i-1} * 1 * Pe_{i+1} * \dots * Pe_{n-1} * Pe_n)] \\ &= S_{\text{gate output}}(AND) * [f(0) \oplus f(Pe_1 * Pe_2 * \dots * Pe_{i-1} * 1 * Pe_{i+1} * \dots * Pe_{n-1} * Pe_n)] \\ &= S_{\text{gate output}}(AND) * [0 \oplus f(Pe_1 * Pe_2 * \dots * Pe_{i-1} * 1 * Pe_{i+1} * \dots * Pe_{n-1} * Pe_n)] \end{aligned}$$

Simplifying it even further yields:

$$Se_i(AND) = S_{\text{gate output}}(AND) * (Pe_1 * Pe_2 * \dots * Pe_{i-1} * 1 * Pe_{i+1} * \dots * Pe_{n-1} * Pe_n).$$

Once the signal probabilities and sensitization probabilities have been estimated, the detection probabilities of a fault x could be estimated by:

$D(x/0) = Px * S(x)$ and

$D(x/1) = (1-Px) * S(x)$.

Where $D(x/0)$ and $D(x/1)$ are the estimated detection probabilities of stuck-at 0 and stuck-at 1, respectively.

Table IV gives a summary of the results for the circuits that were simulated to check the validity of this algorithm. For each fault x the value of $P_p(x)$ is the estimated detection probability by using the proposed algorithm, whereas $P_{SIM}(x)$ is the exact detection probabilities using fault simulator. We indicate in the table how the 'exact' detection probabilities were obtained, i.e., through exhaustive simulation or Monte Carlo techniques. For each circuit, we computed maximum error, average difference, root mean square and correlation coefficient between P_p and P_{SIM}. The table shows high correlation coefficient and small RMS which indicate that the proposed algorithm computations track the exact values very closely. For the circuit C432, an increase in the number of test vectors applied to the circuit (to estimate the exact detection probabilities) resulted in an overall increase in the correlation coefficients.

IV. CONCLUSION

We have presented a practical and effective algorithm based on new set of inference rules for estimating signal probabilities. This algorithm is called the possibilistic algorithm. We have demonstrated that this algorithm is linear and equal to the product of circuit size and the number of primary inputs. We also showed that it provides

TABLE IV
Maximum error, average difference, root mean square and correlation coefficient for some tested circuits

CIRCUIT	Exact_Computation E = Exhaustive M = Monte Carlo	Maximum Error	Average differenc e	RMS of Error	Correlation Coefficient
4-Bit ALU	E	0.40	0.073	0.098	0.917
4-Bit Compartor	E	0.21	0.020	0.074	0.940
9-Bit Parity	E	0.31	0.12	0.114	0.861
C17	E	0.09	0.012	0.036	0.975
C432	M	0.25	0.024	0.075	0.913

significantly better estimates of signal probabilities than other linear algorithms, such as the simple and the weighted averaging algorithm, primarily, on the account of accuracy. Based on this algorithm, the detection probabilities of stuck-at faults are estimated.

ISCAS benchmark circuits were used to validate the proposed algorithm. The experimental results show that reasonable accuracy may be achieved using linear algorithm. Such results were shown to be fairly accurate using a fast linear algorithm.

REFERENCES

[1] K. P. Parker and E. J. McCluskey, "Analysis of Logic Circuits with Faults Using Input Signal Probabilities, " IEEE Transactions on Computers, Vol. C-24, pp. 573-578, May 1975.

[2] K. P. Parker and E. J. McCluskey, "Probabilistic Treatment of General Combinational Networks, "IEEE Transactions on Computers, Vol. C-24, pp. 668-670, June 1975.

[3] R. Garey and D. S. Johnson, Computers and Interactability: A Guide to the Theory of NP-Completeness, San Francisco, CA: Freeman, 1979.

[4] M. Karp and M. Luby, "Monte-Calro algorithms for enumeration and reliability problems," in Proc.IEEE Symp. Fundations Comput. Sci., 1983, pp. 56-64.

[5] Brglez, "On testability analysis of combinational networks. "in Proc. IEEE Symp. Circuits Syst., 1984, pp. 221-225.

[6] Goel, "Test Generation Costs Analysis and Projections,"Proc. 17th Design Automation Conf., June 1980, pp. 77-84.

[7] Savir, G. Ditlow, and P.H. Bardell, " Random Pattern Testability" IEEE, Trans. Comp., vol. C-33, no. 1, pp.79-90, Jan. 1984.

[8] C. Seth, L. Pan, and V. D. Agrawal, "PREDICT - Probabilistic Estimation of Digital Circuit Testability," Fault-Tolerant Computing Symposium (FTCS-15) Digest of Papers, Ann Arbor, MI, pp.220-225, June 1985.

[9] J. Wunderlich, "PROTEST: A tool for probabilistic analysis," in Proc. 22nd Des. Auto. Conf., Las Vegas, June 23-25, 1985, pp. 204-211.

[10] C. Seth, B. B. Bhattacharya, and V. D. Agrawal, "An Exact Analyis for Efficient Computation of Random-Pattern Testability in Combinational Circuits," Fault-Tolerant Computing Symposium (FTCS-16) Digest of Papers, Vienna, Austria, pp. 318-323, July 1986.

[11] krishnamurthy and I. G. Tollis, "Improved Techniques for Estimating Signal Probabilities," IEEE Trans. on Computers, Vol. 38, No. 7, pp. 1041-1045, July, 1989.

[12] Chakravarty and H. Hunt III, " On Computing Signal Probability and Detection Probability of Stuck-at-Faults," IEEE Trans. on Computers, Vol. 39, No. 11, Nov. 1990.

[13] Sreejit Chakravarty and H. B. Hunt III, "On Computing Reliability-Measures of Boolean Circuits, " IEEE Transactions on Reliability, Vol. 40, pp. 582-592, Dec. 1991.

[14] Sami A. Al-Arian and Musaed A. Al-Kharji, "Fault Simulation And Test Generation By Fault SamplingTechniques," 1992 IEEE International Conference on Computer Design, October 1992, pp. 365-368.

[15] H. Debany, K. A. Kwiat and S. A. AL-Arian, " A Method for the Consistent Reporting of Fault Coverage", IEEE 1991 International Work shop on Defect and Fault Tolerance, November 1991.

[16] A. Ali and G. R. Redinbo, "Tight Lower Bounds on the Detection Probabilities of Single Faults at Internal Signal Lines in Combinational Circuits," IEEE Transactions on Computers, Vol. 43, No.12, Dec. 1994.

[17] Vishwani D. Agrawal, H. Farhat, and Sharad C. Seth, "Test Generation by Fault Sampling," Proc. Int. Conf. on Comput. Design (ICCD-88), Rye Brook, NY, Oct.1988, pp. 58-61.

[18] Brgles, H. Fujiwara, "A Neural Netlist of 10 Combinational Benchmark Circuits and A Target Translator in Fortran," IEEE Int'l Symposium on Circuits and Systems, 1985.

[19] Musaed A. Al-kharji and Sami A. Al-Arian, "SATFAST: a StAtistical Tool for FAult Simulation and Test generation," under review.