Timing Yield Calculation Using an Impulse-train Approach

Srinath R. Naidu
Department of Electrical Engineering
Eindhoven University of Technology, The Netherlands

Abstract

This paper presents a new method to compute the probability distribution of the delay of a combinational circuit and uses it obtain an estimate of the yield of the process that manufactures the circuit. We assume a simple delay model assigning a triangular distribution to the delay of a gate and ignore the logical function of the gate and the pin-to-pin delay. The method can handle tree-like circuits as well as circuits with reconvergent fanout in them. The chief advantage of this method over conventional Monte-Carlo simulation is that it is much faster while providing comparable quality.

1 Introduction

As the semiconductor manufacturing process enters the deep submicron era, it is becoming increasingly hard to fix the delay of the circuit. Unlike in the past the statistical variations in the delays of each gate could cause a significant deviation in the mean delay at the output. The deviations in the delays of gates occur due to inter-die and intra-die variations and can be as high as 30 percent of the actual delay of these components [2]. Further the increase in the clock speed limits the allowable deviation in the delay at the output. Since there is a lot of variation in the delays of the individual gates of the circuit, one cannot expect a deterministic timing analysis that works with fixed gate delays to give a realistic idea of circuit delay. It is necessary to obtain a good statistical characterisation of the delay at the output so as to aid the calculation of the number of manufactured circuits that will pass the delay requirement.

The most popular method for obtaining the probability distribution of the delay at the output involves Monte Carlo simulation [2] [5] [6]. However Monte Carlo simulation is expensive as we may needed several thousands of trials to obtain the probability distribution to any reasonable accuracy. In contrast to Monte Carlo simulation some authors have resorted to computing closed-form expressions for the delay distributions at the internal nodes of the circuit, assuming the gate distributions be Gaussian, for example [1]. However, even if one were to assume the delay distributions at the gates to be Gaussian, the analytical expressions for the delays at the internal nodes become complex and time-consuming to compute. In this paper we show the efficacy of a discretisation approach to the problem wherein we discretise the probability distribution of each gate and then propagate these distributions to the output. The discretisation process involves some loss of accuracy but experimental results for tree-circuits appears to indicate that this loss of accuracy is not very significant. The chief advantage is the that the approach is much faster than Monte Carlo simulation. Much like in Monte Carlo simulation the method has a natural framework that allows the user to trade accuracy for computation time. This paper has certain shortcomings. We do not take into account possible dependencies between gate delays due to local effects. For instance, it could be that gates in one part of the circuit are faster than elsewhere in the circuit due to local heating. The delay model we employ assumes that input pin-to-output pin delay is the same for all inputs. This particular shortcoming however can be easily removed by a simple extension of the method in this paper. We do not weed out false paths from the circuit. This is reasonable since it is observed in [5] that there are very few false paths in many practical circuits. Lastly, there is no theoretical analysis of how the errors involved in the discretisation of the probability distributions at the gates of the circuit propagate through the circuit although comparison with Monte Carlo simulation appears to suggest that the errors are very small.

2 Previous Work

Early work in the area of statistical timing analysis consisted of treating the circuit as a PERT network where the nodes in the network are assumed to have delay distributions and the requirement is to find the probability distribution of the delay at the output of the network. In [8], a very general treatment is given where no assumption is made on correlations between node delay random variables. An attempt to establish the probability that each path in the circuit will have a certain timing performance is made in [4]. In [5] the effect of false paths on circuit delay is considered and an attempt is made to factor them in the probability computations. A precise statement of the statistical timing analysis problem is presented in [6]. [1] presents an analytical approach to the problem by making the approximation that the maximum of two normal distributions is a normal distribution. The discretization approach of the probability distributions was proposed by [9] and independently by [7]. The spirit of this paper is very similar to [7] but there are some notable differences in that [7] works with uniform sampling whereas we can work with both uniform and non-uniform sampling. We show why it may become necessary to use non-uniform sampling. Further [7] propose no techniques to control the number of impulses whereas we propose two techniques to do so. Finally, although both approaches use the concept of supergates to handle circuits with reconvergent paths our notations and algorithm are different from [7]. We compare both algorithms. For complex supergates we propose a hybrid solution using both Monte-Carlo simulation as well as our own impulse-train approach to arrive at the delay distribution at the output of the circuit. [7] alludes to a hybrid approach but does not discuss it at length.

3 Delay Model

The arrival time of a signal at the output of gate i can be modelled as

$$Z_i = max(X_i, Y_i) + G_i \tag{1}$$

where X_i and Y_i are the latest arrival times of signals at the inputs of the gate and G_i is the delay of the gate i, and Z_i is the latest arrival time at the output of the gate. This formulation of arrival time of the output essentially means that we assume that any transition on either of the inputs gets transferred to the output and we are calculating the latest time beyond which there are no further transitions at the output (i.e the time at which the output becomes stable). We model the delay of a gate as a random variable obeying a triangular distribution centered around a mean in the interval (mean - a, mean + a) as in Figure 1. Our choice of a triangular distribution is motivated by the fact that the delay of a gate cannot range from -∞ to +∞ but instead must have finite minimum and maximum values, and that the delay distribution must have a single peak. We would like to emphasize that the techniques in this paper can be extended to any distribution. Consider the distribution shown in the figure below in the range (10ns,30ns) centered at 20ns.

In order to discretise the distribution we divide the interval (10,30) into 10 equal parts (10,12),(12,14)...(28,30). We concentrate the probability of the random variable lying in any one of these intervals in an impulse lying at the center of that interval. The height of an impulse is the area

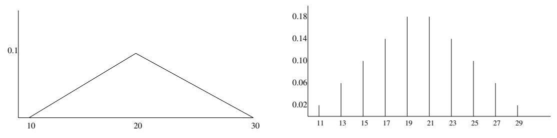

Figure 1. A triangular distribution and its discretised form

of the probability distribution curve over the corresponding interval range. For example the impulse corresponding to the range (10,12) is of height 0.02 which is the probability that the delay lies in the interval (10,12). The idea here is that if one were to divide the interval into a sufficiently large number of sub-intervals, then the discretised distribution will represent in a fairly accurate manner the continuous distribution. Since we calculate a train of impulses to represent the probability distribution of the delay at each gate of the circuit, we call this the “Impulse-train” approach.

4 Computing Delay random variables

In order to compute the delay using the formula of (1) we need to compute the maximum of two random variables followed by a convolution. Let X and Y be discrete random variables with sample spaces $S(X)$ and $S(Y)$ of sizes m and n respectively. Let $S(X) = \{p_i, 1 \leq m\}$ and $S(Y) = \{q_i, 1 \leq n\}$.

Then Z = max(X,Y) is a random variable with size at most m+n, and we can compute the probability that $Z = k$, $Pr(Z = k)$ as follows:

$$Pr(Z = k) = Pr(X < k)Pr(Y = k)$$
$$+Pr(X = k)(Pr(Y < k) + Pr(Y = k)) \tag{2}$$

Here

$$Pr(X < k) = \sum_{p_i < k} Pr(X = p_i) \tag{3}$$

The distribution for the random variable Z = X + Y can be computed by performing a discrete convolution as follows:

$$Pr(Z = k)$$
$$= \sum_{p_i \in S(X)} Pr(X = p_i)Pr(Y = k - p_i) \tag{4}$$

Note that $Pr(Z = k)$ is non-zero only for those values of k that are given by the sum of some pair of values in the sample spaces of X and Y.

It must be noted that the equations for the maximisation and convolution operations are merely the discrete versions of their continuous counterparts. The discretisation of the distributions enables us to carry out the maximisation and convolution operations efficiently for any type of distribution as opposed to an analytical approach that attempts to compute closed-form expressions for the delay. In [1], an analytical approach is proposed assuming that the gate delays are Gaussian and noting that the maximum of two Gaussian random variables is approximately Gaussian. However, this approximation is only valid for certain situations and it is not clear how the distortions will propagate through the circuit.

5 Computational Complexity

Let X and Y are two discrete random variables with sample space sizes $S(X) = m$ and $S(Y) = n$. Then the maximisation operation takes O(mn) time and the the random variable $Z = max(X, Y)$ has a sample space of size O(m+n). Thus the maximisation operation does not cause a blow-up in the sample-space size. However, if $Z = X + Y$, then Z has a sample space size of O(mn) and it takes O(mn) time to compute it. Thus in case of repeated convolution, there is a potential for a blow-up in the number of impulses. In practice, the blow-up does not materialise because the number of distinct values in the sample space of Z is fewer than the theoretical number, mn. In fact, if we were to perform uniform discrete convolution where both sequences to be convolved have the same period, then the number of distinct values in the sample space of Z would actually be of O(m+n). However in a practical setting we are not always able to perform uniform discrete convolution. For instance, if the gate distributions were non-symmetric then it might become necessary to position each impulse at the centroid of the corresponding strip and not at its midpoint in order to ensure that the mean of the discretised distribution is the same as the mean of the original continuous distribution. This would make the sequence non-uniform. For signal lines deep inside the circuit, the delay distribution becomes non-symmetric even if we started with symmetric distributions. Generally these delay distributions rise sharply and fall off slowly due to repeated maximisation of random variables. This is all the more reason to sometimes use non-uniform discretisation at least for blocks deep inside the circuit. One way to get around the problem of a blow-up in the number of impulses is to ensure that the impulses are positioned at the integer value closest to the centroid of the strip. Then the convolution operation would result in output impulses positioned at integer values and the number of impulses would only grow according as the spread of the output distribution defined as the difference between the maximum value of the output random variable and the minimum value of the output random variable. Thus the number of impulses at the output would again be of O(m+n).

Even if we could ensure that the number of impulses at the output of each gate in the circuit is of the order of the sum of the numbers of impulses at the inputs of the gate, we could end up with nodes with a fairly large number of impulses (such as nodes close to the primary output in a circuit with large depth). To reduce the number of impulses for such nodes, we propose two techniques, with contrasting properties, to combine impulses such that the mean of the distribution is preserved and the variance is only slightly changed. In the first technique, we combine the two impulses of height m and n into a single impulse of height $m+n$ located at $d = \frac{mx+ny}{m+n}$, which is merely the "center of mass" of the two impulses. While this operation preserves the mean it does change the variance of the distribution. The change in variance can be computed as follows:

$$\delta v = (\frac{mx + ny}{m + n})^2 (m + n) - (mx^2 + ny^2) \tag{5}$$

After some manipulation, we have $\delta v = -(\frac{mn}{m+n})(y - x)^2$. If we assume that $m > n$, then δv is bounded by $\delta v > -(n(y - x)^2)$. Thus in a practical scenario, we can combine impulses until the penalty in terms of variance is no greater than a small percentage of the original variance. This technique causes a decrease in the variance of the resulting distribution. We propose a second technique that increases the variance of the resulting distribution while reducing the number of impulses. In this technique, the impulse situated at y of magnitude q is combined into the impulses located at x and z such that the mean of the distribution is preserved. To do this, the impulse at x of magnitude p is increased to $p + \frac{q(z-y)}{(z-x)}$ while the impulse at z of height r is increased to $r + \frac{q(y-x)}{(z-x)}$. The change in variance is computed as follows:

$$\delta v = q(z - y)(y - x) \tag{6}$$

Experimental evidence indicates that the combination of these two strategies does result in a reduction in the number of impulses without affecting the final delay distribution too much. However one cannot indiscriminately combine impulses using the above two techniques. An effort should be made to make sure that the resulting impluses are located at convenient values (such as integral values) for efficient further computation.

6 Reconvergent Fanout

Circuits with reconvergent fanout cause the above analysis to become complicated. This is because we lose an important characteristic of fanout-free circuits, namely that the inputs to any given gate are logically independent. A number of techniques have been proposed in the literature to deal with spatial correlations. One such technique is the

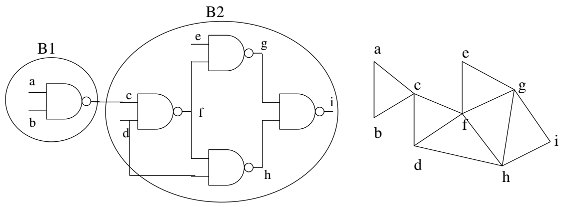

Figure 2. A simple circuit with reconvergent fanout and supergate blocks B1 and B2.

supergate technique first proposed by [10]. The technique consists of first decomposing the original circuit into sub-circuits such that the inputs to each sub-circuit are logically independent. The specific circuit decomposition scheme we use is drawn from [3]. As in [3] we define the supergate of a node in the circuit to be the minimal sub-circuit in the transitive fanin of the node such that the sub-circuit's fanins are logically independent. This technique consists of building a lc-graph corresponding to the given circuit. The vertices of the lc-graph are all the signals of the circuit, and an edge is inserted between two vertices if and only if either the signal lines corresponding to the two vertices are both inputs to some gate or one of them is an input to some gate while the other is the output of the same gate. This procedure has the effect of creating a local clique corresponding to each gate of the circuit. Theorem 1 of [3] establishes that the bi-connected components of the lc-graph are maximal supergates (i.e supergates that are not properly contained in a larger supergate). Figure 2 shows how a simple circuit is partitioned into supergates.

The associated lc-graph of the circuit is shown in the Figure 2. Note that the node c is an articulation point in this graph(i.e vertex whose removal disconnects the graph). Below we present the basic algorithm used to compute the output distribution for a supergate S given in the form of a set of SuperGate expressions.

```
$SuperGateCompute(S, BayesFactor)${
  $Q = RepeatSet(S)$;
  if($Q$ == NULL){
    $S' = Evaluate(S, Q = NULL)$;
    $Scale(SuperGateLocalDist, BayesFactor)$;
    $Combine(SuperGateGlobalDist, SuperGateLocalDist)$;
    $return$;
  }
  if($Q \neq NULL$){
    $T = NewTuple(Q)$;
    $BayesFactor = BayesFactor * P(Q = T)$;
    $S' = Evaluate(S, Q = T)$;
    $SuperGateCompute(S', BayesFactor)$;
    $return$;
  }
}
```

We describe how the above algorithm works in the case of our example. We start with the set of supergate expressions S representing all the gates of the supergate. In our example, for the supergate B2, S is given by $\{f = max(c,d) + G_2, g = max(e,f) + G_3, h = max(f,d) + G_4, i = max(g,h) + G_5\}$ where $G_2 \ldots G_5$ are the gate delay distributions of the respective gates. Given a set of Supergate expressions S, the base set of variables is the set of variables that do not occur on the left hand side of any equation in S. $RepeatSet(S)$ returns a set of variables which occur more than once in the set S but never on the left hand side of any equation in S. For our example set S,the base set is $\{c, d, e\}$ and $RepeatSet(S)$ returns the set $\{d\}$. $NewTuple(Q)$ returns a previously unused tuple from the set of tuples associated with the distributions of the random variables in Q. The set $S' = Evaluate(S, Q = T))$ is got by substituting each variable of Q by the corresponding constant value from T, and then evaluating the set of supergate expressions where possible. This means that if we find the supergate expression of the form $Z = max(X,Y) + G_i$ where X and Y are either base set variables or constants, we can evaluate the expression, and remove it from the list. In our example $S' = Evaluate(S, d = k)$ results in the set $\{g = max(e,f) + G_3, h = max(f,k) + G_4, i = max(g,h) + G_5\}$ as we could remove the supergate expression $f = max(c,k) + G_2$. We perform the above steps in each recursive call until we reach a point where $RepeatSet(S)$ is the empty set. Thereafter, $RepeatSet(S) = NULL$ in successive invocations of SuperGateCompute and we arrive at a point where there is only one expression left S. At this point we can actually evaluate the supergate output. Having done so, we scale the impulse-train corresponding to the supergate output by the computed $BayesFactor$ using the function $Scale(SuperGateLocalDist, BayesFactor)$. We then combine the locally computed distribution with a global distribution maintained at the output. Note that the functions $Scale$ and $Combine$ only perform their tasks when $SuperGateLocalDist$ is available.

The recursion tree for our example is of depth 3. Let us evaluate BayesFactor for one path in this recursion tree. In the first call of SuperGateCompute(S, 1), Q turns out to be the set {d}. The second call to SuperGateCompute sets Q to be {f}. In the third call Q turns out to be NULL. At this point there is only one expression left in S, i.e $i = max(g,h) + G_5$. The BayesFactor used to scale $SuperGateLocalDist$ for the supergate output i is given by $BayesFactor = P(f = T_2/d = T_1) * P(d = T_1)$. The probability magnitude at each sample space point in the impulse-train of i prior to scaling corresponds to the probability $P(i/f = T_2, d = T_1)$. Scaling each impulse in the train by the BayesFactor computed gives us the joint probability $P(i, f = T_2, d = T_1)$ according to Bayes product rule $P(i, f = T_2, d = T_1) = P(i/(f = T_2, d = T_1)) * P(f = T_2/d = T_1) * P(d = T_1)$. Clearly when we are finished with all paths in the recursion tree, we will have computed in the resulting impulse train for the supergate output i, the distribution for i without any dependencies.

7 More Complex SuperGates

For supergates with many stems, the impulse-train approach becomes very impractical and begins to become much more expensive than Monte Carlo simulation. Fortunately we can combine Monte Carlo simulation with the impulse-train approach. The idea here is that we sample the discrete distributions at the inputs of the supergate and perform a Monte-Carlo simulation by sampling the distributions at the gates of the circuit, and collecting the samples at the output. The work of [7] alludes to a hybrid approach but does not discuss how to construct a distribution out of the samples collected at the output. We divide the spread of the distribution into several frequency bins and determine the number of samples that fall into each bin. The probability magnitude for each bin is set to the ratio of the number of samples belonging to that bin to the total number of samples. We locate the probability at the centroid of each bin.

8 Monte Carlo simulation

The traditional approach to statistical timing analysis has been to use Monte Carlo simulation. A crucial issue in the use of Monte Carlo simulation is the number of trials needed to compute the statistical quantity in question. In our case the quantity of interest is the fraction of manufactured circuits that are likely to pass the delay requirement. Let p represent this value. We would like to find the number of trials needed to estimate p with a certain accuracy ϵ at a certain level of confidence $1 - c$. Let $Z_1, ..., Z_n$ be n 0-1 random variables where Z_i =0 or 1 depending on whether the circuit fails or passes the delay requirement.

$$Z = \frac{Z_1 + Z_2 + ...Z_n}{n} \tag{7}$$

It can be shown that Z obeys a binomial distribution with $\mu = p$ and $\sigma^2 = \frac{p(1-p)}{n}$. Using Chebyshev's inequality, we have

$$Pr(|Z - p| \geq \epsilon p) \leq \frac{(\sigma)^2}{(\epsilon p)^2} \tag{8}$$

Substitute for σ^2. In order to have an accuracy of 100ϵ percent with a confidence 100(c) percent, we must have

$$\frac{(1-p)}{np(\epsilon)^2} \leq (1-c) \tag{9}$$

Setting c = 0.99 and ϵ = 0.01 this translates to $n \geq \frac{1-p}{p} \times 10^6$. Thus the number of trials needed for a certain accuracy and confidence in the distribution is dependent on the yield which is the quantity that is sought to be estimated. If we take the yield $p \approx 0.9$ then we will need about 100,000 trials. The interesting thing about this result is that the number of trials for a certain confidence and accuracy needed does not depend on the size of the circuit directly, but instead depends on the yield itself.

9 Results

In tables 1 and 2 below results are given for a lot of randomly generated tree circuits. The notation T-x-y stands for a tree circuit with x levels where each gate has no more than y inputs. The yield is defined here as the precentage of circuits that have a delay that is within 5 percent of the statistical mean. We used 100,000 trials for each Monte Carlo simulation in Table 2. We found that the probability distribution computed by the Impulse-train approach closely tracked the one obtained from Monte-Carlo simulation. However the Impulse-train approach gave more accurate results for circuits with a large number of levels and gates with few inputs as compared to circuits with few levels but with gates having many inputs. This is probably because there is greater error involved in computing the probability distribution of the maximum of several random variables using our method than in computing the probability distribution of the sum of several random variables using our method. In no case was the yield prediction more than 5 percent off the mark. From the table, it is clearly seen that the Impulse train approach is much faster than the Monte-Carlo approach and yields very good results for the circuits considered.

In tables 3 and 4 results are given for a few circuits with reconvergent fanout. These are multi-output circuits, so we do not show mean and variance for each output. Rather we look at the average yield figures for these circuits. As for tree circuits, the yield for an individual output is defined as the probability that the output delay in question is within 5 percent of the statistical mean for that output. Each of these circuits could be decomposed into supergates of reasonable size such that the supergate expressions for the supergate outputs contained only a few inputs that occurred more than once. Attempts are on to extend the method to handle supergates whose supergate expressions have many variables occurring more than once. As can be seen the impulse-train method gave good results for these circuits as well, and in much shorter time.

10 Conclusions

We have presented an impulse-train approach to statistical timing analysis of combinational logic circuits. The ap-

Table 1. Results for Impulse-train method - Tree Circuits

Circuit	Gates	Impulse Method			
		Mean	Variance	Yield	Time(s)
T-3-15	10	62.80	10.08	64.8	0.02
T-4-10	56	87.02	14.61	70.1	0.07
T-8-4	538	180.16	29.71	90.3	1.94
T-9-5	4177	208.98	25.69	95.3	124.99
T-11-4	2645	249.93	42.48	94.5	33.99
T-14-3	3793	315.25	57.58	96.5	44.67
T-15-2	29	313.30	248.222	68	0.47
T-15-3	7897	340.93	48.05	98.5	217.95

Table 2. Results for Monte-Carlo method - Tree Circuits

Circuit	Gates	Monte-Carlo Method			
		Mean	Variance	Yield	Time(s)
T-3-15	10	62.74	9.51	68.3	156.24
T-4-10	56	86.92	14.04	74.9	185.69
T-8-4	538	179.79	28.31	90.1	432.01
T-9-5	4177	208.68	23.57	97	3491.20
T-11-4	2645	249.64	40.37	95.1	1850.54
T-14-3	3793	314.86	55.87	96.7	2111.76
T-15-2	29	313.28	243.05	68.5	142.08
T-15-3	7897	340.59	46.07	98.8	4569.7

Table 3. Results for Impulse-train method - non-Tree Circuits

Circuit	Gates	Impulse-train Method	
		Yield	Time(s)
majority	16	0.58	0.09
i4	402	0.64	4.21
i5	460	0.51	14.75
i9	852	0.62	206.18
comp	211	0.87	0.13
cu	122	0.51	1.64
cm138a	49	0.41	0.46
cordic	205	0.83	2.25
cmb	91	0.71	1.54

proach is shown to be faster than Monte- Carlo simulation. As of now, it can handle circuits with circuits with limited reconvergent fanout. We are trying to improve the method so that it can handle circuits containing more complicated supergates. Efforts are also on to improvise upon the delay model employed making it correspond more closely to reality.

Table 4. Results for Monte-Carlo method - non-Tree Circuits

Circuit	Gates	Monte-Carlo Method	
		Yield	Time(s)
majority	16	0.61	218.22
i4	402	0.64	346.33
i5	460	0.50	345.99
i9	852	0.57	444.44
comp	211	0.84	265.66
cu	122	0.47	238.65
cm138a	49	0.41	221.89
cordic	205	0.84	259.75
cmb	91	0.69	231.5

References

[1] M.Berkelaar "Statistical Timing Analysis, A linear time method", Proceedings of TAU, Austin, TX, December 1997.

[2] R.B.Brashear, N.Menezes, C.Oh, L.T.Pillage, M.R.Mercer, "Predicting Circuit Performance Using Circuit-level Statitsical Timing Analysis", European Design and Test Conference, 1994, pp 332-337.

[3] D.I.Cheng, M.M.Sadowska, K.T.Cheng, "Speeding Up Power Estimation by Topological Analysis", IEEE 1995 Custom Integrated Circuits Conference,pp 623-626.

[4] R.B.Hitchcock, G.L.Smith and D.D.Cheng, "Timing Analysis of Computer Hardware", IBM Journal on Research and Development, pp 100-105, January 1982.

[5] H.F.Jyu, S.Malik, S.Devadas and K.W.Keutzer, "Statistical Timing Analysis of Combinational Logic Circuits", IEEE Transactions on VLSI Systems, Vol.1, No.2, June 1993, pp 126- 137.

[6] R.B.Lin, M.C.Wu, "A New Statistical Approach to Timing Analysis of VLSI Circuits", IEEE International Conference on VLSI Design, pp 507-513, 1997.

[7] J.J.Liou, K.T.Cheng, S.Kundu, A.Krstic, "Fast Statistical Timing Analysis by Probabilistic Event Propagation", Proceedings of 38th Design Automation Conference, June 2001, pp 661-666.

[8] A. Nadas, "Probabilistic PERT", IBM Journal of Research and Development, Vol. 23, No. 3, May 1979, pp 339-347.

[9] S.R. Naidu, "An Impulse-train Approach to Statistical Timing Analysis", Workshop Notes of the International Workshop on Logic Synthesis, June 12-15, 2001, Granlibakken, CA.

[10] S.C.Seth, L.Pan, V.D.Agrawal, "Predict: probabilistic estimation of digital circuit testability," Proceedings of Fault-Tolerant Computing Symposium, 1986.