τAU: Timing Analysis Under Uncertainty

Sarvesh Bhardwaj
ECE Department
University of Arizona
sarvesh@ece.arizona.edu

Sarma B.K.Vrudhula
ECE Department
University of Arizona
sarma@ece.arizona.edu

David Blaauw
EECS Department
University of Michigan
blaauw@umich.edu

ABSTRACT

Due to excessive reduction in the gate length, dopant concentrations and the oxide thickness, even the slightest of variations in these quantities can result in significant variations in the performance of a device. This has resulted in a need for efficient and accurate techniques for performing Statistical Analysis of circuits. In this paper[1] *we propose a methodology based on Bayesian Networks for computing the* **exact** *probability distribution of the delay of a circuit. In case of large circuits where it is not possible to compute the exact distribution, we propose methods to reduce the problem size and get a tight lower bound on the exact distribution.*

1. Introduction

As CMOS technology continues to move further into the nanometer regime, even the slightest of variations in parameters such as gate length, dopant concentrations and oxide thickness can result in significant variations in the performance of a device. The variations cause uncertainty in the circuit performance make it difficult to accurately estimate the yield and forces designers to *over design*, resulting in suboptimal circuits [5].

The conventional methodology to model the effect of variations is to determine the circuit performance assuming for each gate the worst possible value of its delay. This can lead to very pessimistic designs. For example, [14] shows that the worst case delay value for a 16-bit adder can be 30% more than the 3σ delay value. Based on many independent sources of evidence, there appears to be a consensus emerging within the CAD and DA communities that the traditional, deterministic approach to the analysis of circuit behavior (both logical and temporal) will no longer be valid, and probabilistic methods based on stochastic models are more appropriate. An excellent discussion of the sources of uncertainty and the need for stochastic models appears in [10].

Probabilistic Timing Analysis (PTA) is an approach to performing timing analysis where the delays of gates and/or interconnect are *random variables*. The distributions of these individual random variables could be obtained by Monte Carlo SPICE simulation, varying some of the key device and process parameters. In this view, the delay of a circuit is also random variable, but one which is a very complex function of the gate and interconnect delay random variables. The central problem in PTA is to determine the probability distribution of the circuit delay.

PTA is not new and dates back to the mid 1970s, where the focus was on computing the distribution of project completion times in PERT networks [7]. One of the key challenges of PTA is that it involves *maxima* and *sums* of a large number of *dependent* random variables. The first attempt to address the most general version of the problem appears in [13]. If M is a random variable that denotes the arrival time at the circuit output, the solution proposed in [13] is to construct the probability distribution of a new random variable M^* which is *convexly larger* than M. This requires solving a constrained non-linear programming problem of very high dimensionality, and can be computationally prohibitive for modern circuits.

The method proposed in [9] is to reduce the complexity of the underlying problem by obtaining symbolic expressions for the delays of the circuit. Another path based approach has been presented in[12] where the authors start with a set of critical nodes based on static timing analysis. Both the above approaches perform Monte Carlo simulations after initial pruning and can take care of *false paths*. However, the number of paths in a circuit can increase significantly with the circuit size resulting in high complexity.

The approach taken in [3], is to propagate PDFs through the graph, with numerical convolution and multiplication being performed at each step. In the presence of reconvergent paths, random variables are replaced by stochastically larger ones to obtain upper bounds on the PDF. The formulation will result in slightly loose bounds because the pin-to-pin delays of a gate for all fanins are assumed to be independent.

One of the challenges faced with performing PTA is its *computational complexity*. Static timing analysis (STA) can be performed in time and space that is proportional to the circuit size. The complexity of computing the exact probability distribution of the delay of a circuit has been stated to be exponential either in the number of paths [15] or in the circuit size [3], [11] because of the presence of reconvergent fanouts. Even so, exact methods are still of interest as they can be applied to *reduced* circuits and lead to provably good upper or lower bounds.

In this paper we propose a different approach for computing the exact probability distribution of the circuit delay. The approach is based on representing the circuit as a Bayesian Network, which essentially prescribes an efficient method to *factorize* the joint distribution to a *optimal* set of factors. The factorization is made possible by taking advantage of the structural dependencies in the circuit. While the theoretical complexity of this approach is still exponential, unlike other exact methods, it is exponential in the maximum clique size of a graph derived from the circuit, and this maximum clique size grows much slower than the circuit size. We then present several transformations for reducing the size of the circuit and show that this leads to bounds on the PDF. We also present a method to incorporate wire delays in the analysis without considerably increasing the complexity. Note that in our formulation,

[1]This work was carried out at the National Science Foundation's State/Industry/University Cooperative Research Centers' (NSF-S/IUCRC) Center for Low Power Electronics (CLPE). CLPE is supported by the NSF (Grant ♯EEC-9523338 and CCR 0205227), the State of Arizona, and an industrial consortium (http://clpe.ece.arizona.edu)

ICCAD'03, November 11-13, 2003, San Jose, California, USA.

Proceedings of the International Conference on Computer Aided Design (ICCAD'03)
1092-3152/03 $ 17.00 © 2003 ACM

gate delays are assumed to be independent random variables. Correlations between gate delays may arise due to many factors such as processing steps having different effects at different locations on the chip or due to spatial correlations between process parameters [3]. [15] shows that the expection of the maximum of uncorrelated normal random variables is an upper bound on correlated normal random variables and gives an algorithm for finding the expectation of the maximum over all path delays assuming they are uncorrelated.

The organization of rest of the paper is as follows: In section 2, we give the problem formulation of PTA. Section 3 gives a brief introduction to Bayesian Networks and how they are used in our analysis. In section 4 we give various transformations we use for reducing the size of our problem. Section 6 shows how we can include the wire delays without increasing the number of nodes in the circuit. Finally, the experimental results and the Conclusions are given in Section 7 and 8 respectively.

2. Problem Formulation

A logic level netlist C is represented as a Directed Acyclic Graph (DAG) $G = (N, E)$ where the nodes of G correspond to the gates or equivalently gate outputs in C and an edge represents a connection between the corresponding gates in C. Associated with each node in G are two random variables: X_i which represents the arrival time of the output signal at that gate, and D_i which represents the delay of the gate. We assume that the gate delays are bounded and constitute a finite set of alternatives.

Let X_i be a node in G with delay D and with inputs from nodes labeled $X_{i_1}, X_{i_2}, \ldots X_{i_k}$. Then

$$X_i = max\{X_{i_1}, X_{i_2}, \ldots, X_{i_k}\} + D \quad (1)$$

We want to find the distribution of arrival time of the primary outputs $O_1, O_2, .., O_m$. The distribution of any signal X_i in the circuit is given in terms of its fan ins $(X_{i_1}, X_{i_2}, \ldots, X_{i_k})$. The arrival times of the fanins are not independent because of presence of reconvergent fanins. Hence finding a closed form expression for $P(X_i \leq t)$ is not possible. Traversing all the way back to primary inputs will result in the arrival time at the circuit output being represented in terms of the arrival times of all the gates in the circuit. Thus it seems that to compute the probability distribution of the max of the outputs will require us to first compute the joint distribution of the arrival times of all the nodes in the circuit. The space required for such a computation will be exponential in the circuit size. However, in the next section we show that through the use of Bayesian Networks, the computation of the joint distribution is not necessary.

3. Introduction to Bayesian Networks

In this section we explain how we can utilize Bayesian Networks (BNs) to obtain the exact probability distribution of a node in a DAG. BNs were introduced to circuit analysis by [4] to compute the switching activity of the signals.

DEFINITION 3.1 ([8]). *A* **Bayesian Network** *is a set of variables and a set of directed edges between the variables which form a directed acyclic graph (DAG). Each variable A has a finite number of mutually exclusive states which it can take and if $B_1, B_2, .., B_n$ are its parents then we associate a conditional probability distribution $P(A/B_1, B_2, .., B_n)$ with that node.*

From the definition above, we see that our representation of the circuit is a Bayesian Network.

DEFINITION 3.2. *A graph is called* complete *if every pair of vertices are joined. A* clique *is a maximal complete subgraph.*

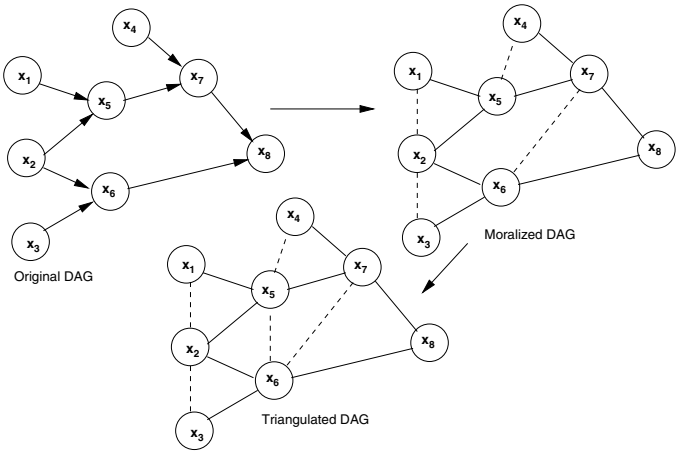

Figure 1: Moralization and Triangulation of a DAG

Consider the DAG shown in Figure 1. The classical approach for obtaining the probability distribution of the output X_8 is to compute the joint probability distribution of $X_1, X_2, \ldots X_8$ as $P(X_1, X_2, \ldots X_8)$ and then compute $P(X_8)$ as

$$P(X_8) = \sum_{X_1, \ldots X_7} P(X_1, X_2, \ldots X_8)$$

The space complexity of this approach is $O(m^n)$ where m is the number of distinct values taken by each variable and n is the number of variables . Moreover it is not clear how to obtain this joint distribution. However, there exists an efficient way of computing the probability distribution of X_8 by factoring the joint distribution and performing efficient marginalization of these factors. For example, $P(X_8)$ can be obtained by

$$\sum_{X_7, X_8} P(X_8/X_7, X_6) \sum_{X_5} \sum_{X_4} P(X_7/X_5, X_4) P(X_4) \sum_{X_2} P(X_2) \cdot \sum_{X_1} P(X_5/X_2, X_1) P(X_1) \sum_{X_3} P(X_6/X_3, X_2) P(X_3) \quad (2)$$

From the above equation we see that we have scheduled the marginalizations such that we don't have to compute the joint probability distribution of more than 3 variables at any time.

The method is based on separating the nodes in the DAG into different subsets such that the joint distribution of the nodes in a subset can be computed and the marginal distribution of any node can be obtained from these joint distributions. The DAG is first converted to an undirected graph by replacing the directed edges with undirected edges. Since the distribution of any node can be determined given the distribution of its parents, the node and its parents should lie in the same set. To ensure this, the graph is moralized by connecting the parents of each node. To obtain the ordering in which to perform the marginalizations as shown in (2), we need to triangulate the graph and remove any chordless cycle of length greater than 3. The moralized and triangulated graphs along with the original DAG are shown in Figure 1.

From the triangulated graph we can obtain different cliques. The subgraph on nodes X_6, X_7 *and* X_8 in the Triangulated DAG in Figure 1 forms a clique, whereas X_5, X_6, X_7 *and* X_8 does not because the edge from X_5 to X_8 is absent. Bayesian Networks help us to partition the circuit into different *cliques* so that we can obtain the distribution of a node N_i by computing the joint distribution of nodes in a clique C_i such that $N_i \in C_i$. These cliques represent the sets over which we have to compute the joint distribution. Using this triangulated graph, we construct a clique tree as shown in Figure 2. An

Proceedings of the International Conference on Computer Aided Design (ICCAD'03)
1092-3152/03 $ 17.00 © 2003 ACM

Table 1: Maximum clique size in benchmark circuits

Circuit	Nodes	Edges	Max. Clique size	No. of Cliques
C17	11	12	4	8
C880	443	729	53	305
C432	196	336	38	150
C499	243	408	32	183
C1355	587	1064	49	402
C1908	913	1497	67	678
C2670	1426	2075	89	1084
C3540	1719	2936	189	1195
C5315	2485	4386	139	1701
C7552	3719	6144	77	2593

edge between two cliques represents that there are common variables between the two cliques. The details of the entire procedure are beyond the scope of this paper, however the detailed algorithms for performing each of these steps and their proofs are given in [17, 8, 6]. The probabilities of the inputs X_1 and X_2 are assigned to clique 1 whereas that of X_3 is assigned to clique 2. Using Bayes theorem, the distribution of C_1 is given by

$$P(X_1,X_2,X_5) = P(X_5/X_1,X_2)P(X_1)P(X_2)$$

we assume that the input arrival times are independent of each other or their joint distribution is given to us. Thus we can compute the distribution of C_1 we can apply the same procedure for C_2. After this, we can obtain the distribution of C_3 as follows

$$P(X_6,X_5,X_2) = (\sum_{X_1}\phi_{C_1})(\sum_{X_3}\phi_{C_2}) = P(X_6/X_2)P(X_5,X_2) \qquad (3)$$

where ϕ_{C_1} is $P(X_5,X_2,X_1)$ and ϕ_{C_2} is $P(X_6,X_3/X_2)$. This shows that the probability distribution of clique 3 can be obtained from that of cliques 2 and 1. Following the same procedure, we can obtain the joint distribution of the variables in clique 6 and obtain the marginal distribution of X_8 from that. The complexity of this procedure is $O(m^c)$, where c is the size of the largest clique. Thus Bayesian Networks can be seen as an efficient tool for computing the distribution of any variable in the Network by dividing the network into smaller subsets and by computing the joint distribution over these subsets. As is evident from this example, the complexity is directly related to the maximum clique size of the DAG.

The maximum clique size present in a BN is dependent on the amount of reconvergence in the network as well as the maximum fanin in the circuit. Since the maximum fanin in a circuit is bounded (typically 10-15), the clique sizes will be much smaller than the circuit size. Moreover, the clique sizes also depend on the quality of heuristics used for the triangulation algorithm. Thus by using a better triangulation algorithm, we can further reduce the clique sizes. Fourth column in Table 1 shows the maximum clique sizes for the ISCAS85 benchmark circuits thus confirming that the clique size is a slow growing function of the circuit size.

To specify the Bayesian Network we construct the conditional probability distributions (CPDs) $P(Y/X_j,X_{j+1},..,X_k)$ for each of the nodes (Y) in the circuit as follows

$$\begin{aligned} P(Y/X_j,X_{j+1},..,X_k) &= \sum_d P(Y,D=d/X_j,X_{j+1},..,X_k) \\ &= \sum_d P(Y/X_j,X_{j+1},..,X_k,D)P(D=d) \end{aligned}$$

4. Graph Transformations

The complexity of constructing the exact *PDF* of the circuit delay can be reduced either by devising new methods to reduce the circuit size and obtain tight lower bounds on the probability distribution (*PDF*) of the delay. There are two main aspects of the problem where the reduction can be performed.

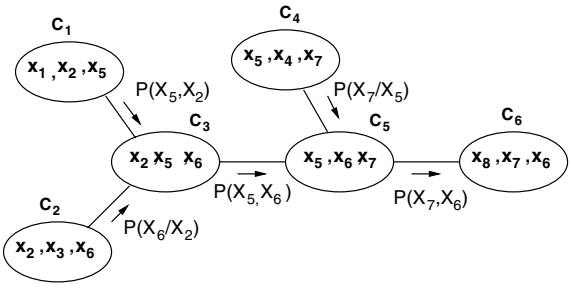

Figure 2: Clique tree of example circuit

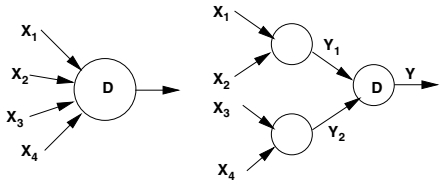

Figure 3: Reducing the Maximum Fanin in the DAG

1. Reduce the graph size based on transformations which will remove/combine nodes,
2. Reduce the size of CPDs in case of nodes with large fanins.

In this section we present a set of results which can be used for performing transformations for reducing the complexity of the analysis. The proofs for all these transformations can be obtained in the extended version [2].

4.1 Fanins Reduction

The size of the conditional probability distribution depends exponentially on the number of fanins of a node. Hence if a node has k fanins where each one of them can take m distinct values, then the size required to store this distribution is $O(m^{k+1})$, an additional dimension for the output of the node. Since in the original circuit, we can have a gates with large fanins (C5315 has a maximum fanin of 9), the size of the largest CPD will be $O(m^9)$. This size is extremely large even for small m, hence we break the node as shown in Figure 3. The delay associated with each of the new nodes is 0, whereas the last node has delay of the original node associated with it. Hence the complexity of storing the CPTs will be $O(m^3)$.

4.2 Reducing Switching Window Size

We are interested in computing the distribution of the arrival times over the switching window of the output. For yield analysis purposes, typically we are interested in the distribution close to the Latest Arrival Time (LAT). Hence by removing some events which result in an arrival time close to the Earliest Arrival Time (EAT) of the outputs, we can prune out a significant number of nodes without sacrificing accuracy. The circuit size can be reduced using this idea by propagating a critical time (T^*) from the output to the primary inputs just as required time is propagated in STA. In this section we prove that by reducing the graph using this transformation results in obtaining a lower bound to the distribution of the circuit delay.

DEFINITION 4.1 ([18]). *If X and Y are two random variables with sample spaces S_X and S_Y, $S_Y \subseteq S_X$, then Y is* stochastically larger *(s.l.) than X, denoted by $Y \geq_{st} X$, if and only if*

$$P(Y > t) \geq P(X > t) \quad \forall\, t \in S_Y$$

Proceedings of the International Conference on Computer Aided Design (ICCAD'03)
1092-3152/03 $ 17.00 © 2003 ACM

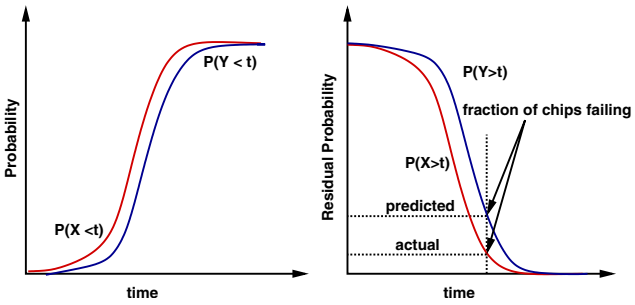

Figure 4: Probability and Residual Probabilities

Figure 4 shows that if Y is stochastically larger than X, then the probability distribution of Y is a lower bound on the probability distribution of X. Hence if we estimate the fraction of circuits whose delay will be greater than a critical value t (i.e. $P(X > t)$), we will never underestimate this fraction if we replace X with a stochastically larger random variable Y.

In the following the term *reduced DAG* means the DAG obtained by reducing the switching window size. Let Y_{red} be an output node in the reduced DAG and let Y be the corresponding node in the original DAG. The following sequence of results are aimed at demonstrating that $Y_{\text{red}} \geq_{\text{st}} Y$.

Let X' be a primary input in the reduced DAG and let X be the corresponding input in the original DAG. Let l_X^* be the required time associated with X as a result of propagating a critical time T^* from the primary outputs to the primary inputs. Then $X' = \max\{X, l_X^*\}$. Since $\{X' \leq t\} \equiv \{X \leq t, l_X^* \leq t\} \subseteq \{X \leq t\}, X' \geq_{\text{st}} X$. Thus all the inputs in the reduced DAG are stochastically larger than the corresponding inputs in the original DAG.

Now consider an arbitrary node Y' in the reduced graph and let Y be corresponding node in the original DAG. After performing the *Reduce Fanin* transformation, either the two inputs to Y' originate from the same node as shown in Figure 5, or they are independent. In either case that Y' is stochastically larger than Y.

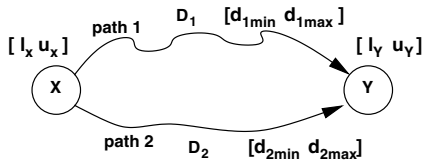

Figure 5: Dependencies between the fanins

LEMMA 4.1. *Let X and Y be two random variables, with sample space $[l_X, u_X]$ and $[l_Y, u_Y]$ respectively, such that $Y = X + max\{D_1, D_2\}$ where D_1 and D_2 are two other random variables with sample space $[d_{1min}, d_{1max}]$ and $[d_{2min}, d_{2max}]$ respectively. Also let X' and Y' be two random variables, with sample space $[l_X^*, u_X]$ and $[l_Y^*, u_Y]$ respectively, such that $X' = max\{X, l_X^*\}$, and $Y' = X' + max\{D_1, D_2\}$ then $Y' \geq_{st} Y$.*

So far we have shown that the primary inputs of the reduced DAG are stochastically larger than the primary inputs in the original DAG. We have also shown that if two paths from a signal which is *s.l.* than the corresponding signal in the original DAG reconverge at some node, then the signal at the point of reconvergence will also be stochastically larger. To complete the validity of this transformation, we need to show that the signals at second level are also *s.l.* than the signals at first level in the original DAG.

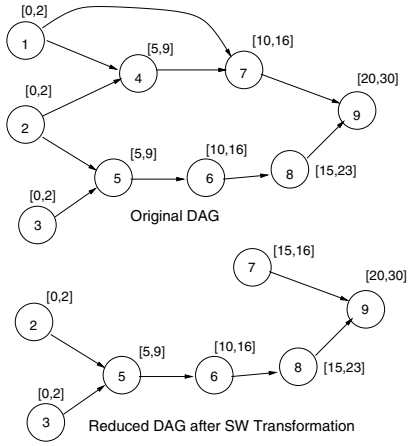

Figure 6: DAGs before and after SW Transformation

THEOREM 4.1. *Let X_1 and X_2 be the random variables denoting the arrival times of two primary inputs of a circuit. Let X_1' and X_2' represent the random variables denoting the arrival times of the same primary inputs in the reduced DAG obtained by assigning a required time T^* at the outputs. Also let Y be represent an RV such that $Y = max\{X_1, X_2\} + D$ and Y' be the corresponding arrival time in the reduced DAG, then assuming the input arrival times to be statistically independent, $Y' \geq_{st} Y$.*

Hence the signals at level 2 (whose fanins are primary inputs) in the reduced DAG are *s.l.* than the corresponding signals in the original DAG. In a circuit, the only dependencies present are those caused by reconvergences. Hence for any signal, either its fanins are independent or the dependency is as described in Lemma(4.1). Since, the signals at level 2 in the reduced DAG are *s.l.* than the signals in original DAG, the signals at subsequent levels will also be stochastically larger.

Figure 6 shows the original DAG and the reduced DAG by taking the critical time T^* to be the EAT of the output. The critical time at the input of each node is obtained by subtracting d_{min} of the node from its critical time. If there is an internal node whose all fanins are removed (e.g. Node 7 in Figure 6), the *Latest Arrival Time* of that node is assigned a Probability of '1' to ensure that it is *s.l.* than the same node in Original DAG.

4.3 Inputs Reduction

Depending on the relative alignment of the switching windows of the two inputs to a node, we can perform further reduction in the graph size.

THEOREM 4.2. *If A and B are the fanins of a node C such that the latest arrival time of A is $\leq$ than the earliest arrival time of B, then we can remove A from the fanin of C without affecting its probability distribution of C.*

Note that [3] also uses this transformation. We can remove A from the entire graph only if all fanouts from A which satisfy the above condition. We can also remove all the nodes in the fanin cone of A, if none of the nodes fanout to a node outside this cone.

4.4 Series Reduction

We also perform a much widely known form of transformation which reduces the complexity of analysis by combining two nodes as shown in Figure 7.

Let the delays of the two nodes be D_1 *and* D_2 respectively. The delay of the reduced node is $D = D_1 + D_2$. Since we have as-

Proceedings of the International Conference on Computer Aided Design (ICCAD'03)
1092-3152/03 $ 17.00 © 2003 ACM

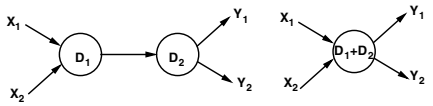

Figure 7: DAG before and after Series Reduction

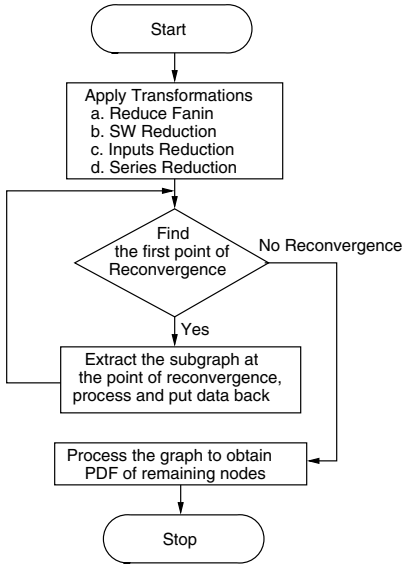

Figure 8: Flowchart for the entire process

sumed that there is no correlation between the gate delays, from general probability theory [16], we know that the probability density of the sum of two independent random variables is obtained by convolving their probability density functions.

5. Process Flow

Figure 8 shows the flowchart corresponding to the steps involved in our analysis. We start the analysis by performing the different transformations in the given order. After reducing the DAG, we traverse nodes in topological order to find a node which is the point of reconvergence. If we find such a node, we extract this node along with its fanin cone. We compute the CPDs for all the nodes in this subgraph and perform the BN-based analysis to compute the PDFs of all the nodes in this subgraph.

After this analysis, we can remove all the nodes which don't fanout to a node outside this fanin cone. Also since we have computed the PDF of the node at which the paths reconverge, we can remove its fanin edges. This will reduce the complexity of the remaining subgraph. But this procedure amounts to making this node independent of all the nodes in its fanin cone and can result in reducing the quality of the bound.

Once we have analysed all the subgraphs containing reconvergences, we perform the analysis on the remaining DAG and obtain the PDF of the output.

6. What about Wire Delays?

Even though a significant amount of work has gone into development of PTA tools, the problem of including wire delays and still performing the analysis efficiently has not been discussed in detail. With the design process going into nanometer scale, the interconnect delays have started to increase sharply primarily because thinner wires result in higher resistance and their close proximity results in higher capacitance. This leads to a higher *RC* value and thus larger delay.

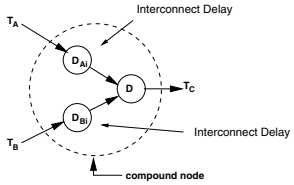

Figure 9: Compound Node for Wire Delays

A simple way to include the wire delays is to insert a node on each of the edges of the DAG, however this procedure will increase the DAG size by the number of edges in the DAG. This number can be as high as twice the number of nodes, hence it will significantly increase the size of the DAG. In this section we present an efficient way of including wire delays, without increasing the number of nodes in the DAG along with only a minor increase in the memory space.

We first insert dummy nodes corresponding to the interconnect delays on each of the edges as shown in Figure 9. Let us denote by D_{A_i} and D_{B_i} the interconnect delays corresponding to the edges connecting A to C and B to C respectively. Let T_A, T_B and T_C represent the arrival times at A, B and C respectively. We assume that the gate and interconnect delays are independent, thus delay random variables D_{A_i}, D_{B_i} *and* D are independent of each other. To keep the DAG size the same, we now combine the three nodes into a *compound node*. To carry on with our analysis, we only need the CPD corresponding to this *compound node*. Hence we compute the CPD of this node as follows.

The delay at the output C is given by

$$T_C = max\{T_A + D_{A_i}, T_B + D_{B_i}\} + D$$

The probability distribution of T_C conditioned on T_A and T_B, $P(T_C = t/t_A, t_B)$ can be computed as follows

$$\sum_{d, d_{A_i}, d_{B_i}} P(max\{t_A + d_{A_i}, t_B + d_{B_i}\} = t - d/t_A, t_B, d, d_{A_i}, d_{B_i}) \bullet$$
$$P_D(d) P_{D_{A_i}}(d_{A_i}) P_{D_{B_i}}(d_{B_i}) \quad (4)$$

Hence, given a particular t_A, t_B *and* t_C, we can obtain the corresponding value of the conditional probability of T_C with respect to T_A *and* T_B from (4). We see that the number of nodes in the DAG remains the same but there is a slight increase in the complexity because the number of distinct values taken by each of the signal (m) increases.

7. Experimental Setup and Results

We performed our analysis on ISCAS85 benchmark circuits and compared our results with 10,000 runs of Monte Carlo simulations. The delays of the gates were mapped using a user specified library for assigning different delay distributions depending on the gate type and the fanins/fanouts. We ran our simulations on a Sun 280r server having 2 Sparc III processors 900 MHz, and 4 GB RAM.

The *Reduce fanin* and *Switching Window Reduction* transformations were implemented in PERL and the resulting DAG was given as an input to MATLAB program. The *Inputs Reduction* and *Series Reduction* transformations and the remaining procedures were implemented in MATLAB. The BN-based analysis was done using *Bayesian Network Toolbox* in MATLAB [1].

Table 2 shows the reduction in gate sizes we obtain after performing Switching Window and Series Reduction transformations. We can obtain as much as 90% reduction in the circuit sizes and the average reduction obtained was 71%.

Proceedings of the International Conference on Computer Aided Design (ICCAD'03)
1092-3152/03 $ 17.00 © 2003 ACM

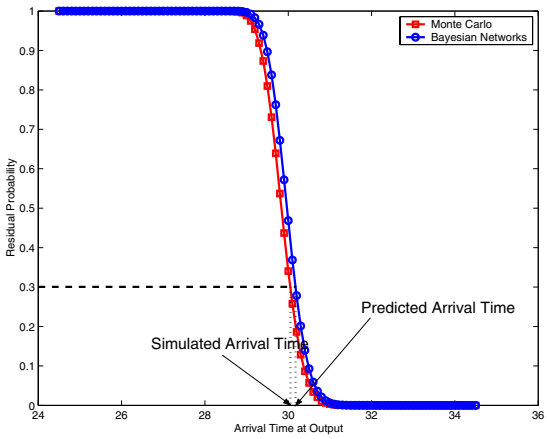

Figure 10: Sim. and Predicted Arrival times for C5315

Table 2: Reduction in the problem size

Circuit	Nodes	Nodes Remaining after		% red.
		Swit. Win.	Series red.	
C17	11	7	7	36%
C432	196	71	29	85%
C499	243	222	158	35%
C880	443	118	51	88%
C1355	587	494	446	24%
C1908	913	179	107	88%
C2670	1426	416	236	83%
C3540	1719	252	165	90%
C5315	2485	379	251	90%
C7552	3719	660	469	87%

We took the percentage variation in the delay $((d_{max} - d_{min})/d_{mean})$ to range from 20 - 40%. The SW reduction was performed with the critical time as the EAT of the output. The EAT was propagated to the primary inputs by subtracting d_{min} of each gate encountered in the path. The amount of reduction using the *Switching Window* transformation depends on the percentage variation in the delay.

We could obtain the exact distribution for C17. Because of large number of reconvergences present in other circuits we obtained their bounds. In Table 3 we see that the worst case difference in the simulated (MC) and predicted (BN) 3σ values is less than 3%. The runtime of our procedure was significantly less than the Monte-Carlo simulations.

Figure 10 shows the simulated and predicted arrival time of C5315 for residual probability of 0.3. We see that the predicted residual probability is always greater than the simulated residual probability. Hence we never underestimate the fraction of circuits whose delay is greater than a given critical time.

Table 3: Comparison of TAU and Monte Carlo simulations

Circuit	$\mu+3\sigma$		Diff.(%)	Time	
	MC	BN		MC	BN
C17	3.54	3.54	0	7.73	0.51
C432	18.07	18.1	0.22	239	89
C499	12.04	12.05	0.05	297	63
C880	25.36	25.40	0.155	516	90
C1355	25.63	25.71	0.38	707	496
C1908	26.18	26.25	0.276	1147	288
C2670	33.70	34.0	0.88	1625	92
C3540	44.5	45.8	2.834	2313	212
C5315	31.1	31.2	0.32	3782	150
C7552	27.3	27.4	0.365	4749	696

Since the computations have been performed in MATLAB, the runtime of our analysis is much slower than if we had a Bayesian Networks package implemented $C/C++$. Hence significant speedups can be obtained by implementing the code in C.

8. Conclusions

We introduced a new methodology for performing PTA of circuits. We showed that the problem of finding exact probability distribution of the arrival time of a signal in the circuit is exponential in the maximum clique size of a graph derived from the circuit. We presented various analytical results using which we performed different graph transformations to reduce the problem size without having significant effect on the accuracy. We introduced a method for incorporating wire delays in the analysis without significant increase in the complexity. Our transformations can result in as much as 90% reduction in the circuit size with the average reduction being 71%. Also the maximum difference in the computed 3σ values and the simulated 3σ values is less than 3% which shows the accuracy of the approach.

9. REFERENCES

[1] http://www.ai.mit.edu/~murphyk/software/bnt/bnt.html.

[2] http://www.ece.arizona.edu/~sarvesh/pubs.html.

[3] A. Agarwal, D. Blaauw, V. Zolotov, and S. Vrudhula. Statistical timing analysis using boounds and selective enumeration. In *Workshop TAU*, pages 332–337, Dec 2002.

[4] S. Bhanja and N. Ranganathan. Dependency preserving probabilistic modeling of switching activity using bayesian networks. In *Proc. of DAC*, pages 209–214, 2001.

[5] D. Boning and S. Nassif. *Models of Process Variations in Device and Interconnect: In Design of High-Performance Microprocessor Circuits*. A. Chandrakasan, 2000.

[6] R. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Springer, 1999.

[7] S. E. Elmaghraby. *Activity Networks: Project Planning and Control by Network Models*. John Wiley & Sons, 1977.

[8] F. V. Jensen. *Bayesian Networks and Decision Graphs*. Springer, 2000.

[9] H.-F. Jyu, S. Malik, S. Devdas, and K. Keutzer. Statistical timing analysis of combinational logic circuits. *IEEE Trans. on VLSI*, 1(2):126–137, June 1993.

[10] K. Keutzer and M. Orshansky. From blind certainty to informed uncertainty. In *Workshop TAU*, pages 37–41, Dec 2002.

[11] J.-J. Liou, K.-T. Cheng, S. Kundu, and A. Krstic. Fast statistical timing analysis by probabilistic event propagation. In *Proc. of DAC*, pages 661–666, 2001.

[12] J.-J. Liou, A. Krstic, W. L.-C., and K.-T. Cheng. False-path-aware statistical timing analysis and efficient path selection for delay testing and timing validation. In *Proc. of DAC*, pages 566–569, 2002.

[13] A. Nadas. Probabilistic pert. *IBM J. Res. Develop*, pages 339–347, May 1979.

[14] A. Nardi, A. Neviani, E. Zanoni, M. Quarantelli, and C. Guardiani. Impact of unrealistic worst case modeling on the performance of vlsi circuits in deep submicron cmos technologies. *Semiconductor Manufacturing, IEEE Transactions on*, 12(4):396–402, Nov 1999.

[15] M. Orshansky and K. Keutzer. A general probabilistic framework for worst case timing analysis. In *Proc. of DAC*, pages 556–561, 2002.

[16] A. Papoulis. *Probability, random variables, and stochastic processes*. Mc-Graw Hill, 1965.

[17] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[18] S. M. Ross. *Stochastic processes*. Wiley, 1996.

Proceedings of the International Conference on Computer Aided Design (ICCAD'03)
1092-3152/03 $ 17.00 © 2003 ACM