Probability Distribution of Signal Arrival Times Using Bayesian Networks

Sarvesh Bhardwaj, *Student Member, IEEE*, Sarma Vrudhula, *Senior Member, IEEE*, and David Blaauw, *Member, IEEE*

***Abstract*—This paper presents a new method based on Bayesian networks (BNs) for computing the exact probability distribution of the delay of a circuit. The method is based on BNs, which allows an efficient means to factor the joint probability distributions over variables in a circuit graph. The space complexity of the method presented here is $O(m^{|C|})$, where m is the number of distinct values taken by each delay variable and $|C|$ is the number of variables in the largest clique. The maximum clique size present in a BN is shown to be much smaller than the circuit size. For large circuits, where it is not practically feasible to compute the exact distribution, methods to reduce the problem size and get a lower bound on the exact distribution are presented. Comparison of the results with Monte Carlo simulations shows that we can reduce the size of the circuit by as much as 89% while maintaining the maximum difference between the predicted and simulated 3σ values to be less than 3%.**

***Index Terms*—Bayesian networks, probability, process variations, timing, yield estimation.**

I. Introduction

As COMPLIMENTARY metal–oxide–semiconductor (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 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. Hence, compared to the traditional deterministic approach to analyze circuit behavior (both logical and temporal), probabilistic methods based on stochastic models are more appropriate [1]. An excellent discussion of the sources of uncertainty and the need for stochastic models appears in [2].

Probabilistic timing analysis (PTA) is an approach to performing timing analysis where the delays of gates and/or interconnect are *random variables*. In this view, the delay of a circuit is also a random variable, but one that is a very complex function of the gate and interconnect delay random variables. The central problem in PTA is determining the probability distribution of the circuit delay.

A difficulty that arises in PTA is that it involves *maxima* and *sums* of a large number of *dependent* random variables. If M is a random variable that denotes the arrival time at the circuit output, the solution proposed in [3] is to construct the probability distribution of a new random variable M^* such that the *expected tardiness* of M^* at time t $(E[(M^* - t)])$ is always larger than the *expected tardiness* of M at time t for all t. This requires solving a constrained nonlinear programming problem of very high dimensionality and can be computationally prohibitive for modern circuits.

The method proposed in [4] 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 [5] that starts with a set of critical nodes based on static timing analysis. Both of 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 [6] is to propagate discrete probability distribution functions (*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 (s.l.) ones to obtain upper bounds on the *pdf*. The method described in [7] computes the probability distribution of the circuit delay from the joint distribution of the parametric variations by an efficient scheme that performs the integration over the feasible region.

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 [8] or in the circuit size [6] because of the presence of reconvergent fan-outs. 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.

Another form of correlation between the arrival times occurs because of the spatial correlations between the underlying sources of variation such as the length of the gate, its threshold voltage, etc. Various approaches that take into account this kind of correlation have been proposed in [9] and [10].

In this paper, we present a different approach for computing the exact probability distribution (over the quantization errors introduced in converting a continuous distribution to a discrete distribution) of the circuit delay in the presence of path reconvergence. The approach is based on representing the

Manuscript received April 29, 2004; revised August 1, 2004. This work was carried out at the National Science Foundation (NSF) Center for Low Power Electronics (CLPE) under Grant EEC-9523338 and NSF Information Technology Research (ITR) Grant CCR-0205227. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. This paper was recommended by Associate Editor S. Sapatnekar.

S. Bhardwaj and S. Vrudhula are with Arizona State University, Tempe, AZ 85281 USA (e-mail: sarvesh.bhardwaj@asu.edu; vrudhula@asu.edu).

D. Blaauw is with the University of Michigan, Ann Arbor, MI 48105 USA.

Digital Object Identifier 10.1109/TCAD.2005.852436

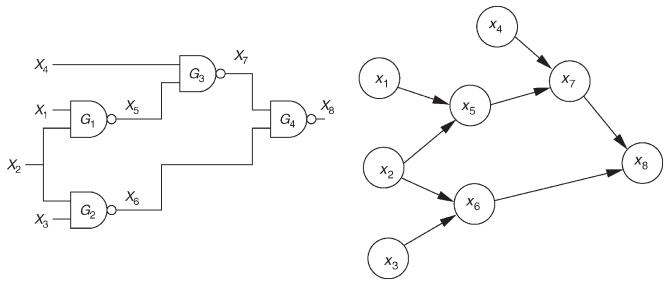

Fig. 1. Modeling of the circuit as a DAG.

circuit as a Bayesian network (BN) [11]. 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. In the cases where it is not possible to compute the exact distribution, we present several transformations for reducing the size of the circuit and show that this leads to good bounds on the *pdf*. Note that in our formulation, gate delays are assumed to be independent random variables but path delays are not. In [8], it is shown that the circuit delay in the absence of spatial correlations between the gate delays is a lower bound to the delay in the presence of correlations. Thus, treating the gate delays as independent random variables is conservative. BNs have been used earlier in the circuit analysis to efficiently compute the switching activity of the signals [11]. This paper gives a detailed description of the concepts introduced in [12] and provides detailed results to emphasize the importance of circuit transformations to reduce the problem size.

The organization of rest of the paper is as follows. Section II contains the formulation of PTA. Section III gives a brief introduction to BNs and how they are used in our analysis. Section IV presents several transformations for reducing the size of the problem. Section V shows how wire delays can be included without increasing the number of nodes in the circuit. Finally, the experimental results and the conclusions are given in Sections VI and VII, respectively.

II. Problem Formulation

A logic level netlist (e.g., Fig. 1) 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. Fig. 1 shows the modeling of a circuit in the form of a DAG. It should be noted that in practice the gate delays are continuous variables. Hence, the gate delays need to be discretized over their respective ranges. Thus, each node X has a minimum delay $d_{X\min}$ and a maximum delay $d_{X\max}$ associated with it and the delay random variable has a discrete distribution over this range.

Let X be the arrival time of a node in G with delay D and inputs from nodes having arrival times as $X_1, X_2, \ldots, X_k$. Then

$$X = \max\{X_1, X_2, \ldots, X_k\} + D. \tag{1}$$

The objective here is to compute the *pdf* of the arrival times of the primary outputs $O_1, O_2, \ldots, O_m$. This representation of a combinational circuit in the form of a DAG is similar to the representation in [11], the only difference being that the variables in this case represent the signal arrival times instead of switching activity. This work computes the distribution of the signal arrival times under the *max* delay model (1); hence, it is suitable for analyzing the setup time violations. For the hold time violations, a *min* delay model can be used and results similar to the ones outlined in this work can be derived.

The distribution of the arrival times of any node X in the circuit is given in terms of the arrival times at its fan-ins $(X_1, X_2, \ldots, X_k)$. The delay D of the gate is independent of the arrival times of the fan-ins, but because of the presence of reconvergent fan-ins, the arrival times of the fan-ins are not independent. Hence, finding a closed form expression for $P(X \leq t)$ 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 arrival times at the outputs will require computing the joint *pdf* 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 BNs the computation of the joint *pdf* is not necessary. There are other approaches that do not compute the joint *pdf* of the arrival times of the nodes in the circuit [6]. We will provide a comparison of our approach with this approach with an example. Other approaches such as propagating cutsets of the circuit from primary inputs to primary outputs could also be used, but these approaches also have very high computational complexity.

III. Introduction to BNs

Definition 3.1 [13]: A BN is a set of variables and a set of directed edges between the variables that form a DAG. Each variable A can assume one of a finite set of states, and if $B_1, B_2, \ldots, B_n$ are its parents, then we associate a conditional probability distribution $P(A/B_1, B_2, \ldots, B_n)$ with A.

In [11], it was proved that the representation of the circuit in the form of a DAG structure is a BN. This is true in the present case as well. Each node in the BN corresponds to the random variable representing the signal arrival time at that node. For each node X, a probability distribution of the signal arrival time at X can be defined conditioned on the arrival times at the inputs of X.

Consider the DAG shown in Fig. 1. The brute force approach for obtaining the *pdf* of X_8 is to compute the joint *pdf* of $X_1, X_2, \ldots, X_8$ as $P(X_1, X_2, \ldots, X_8)$ and then compute $P(X_8)$ as $P(X_8) = \sum_{X_1,\ldots,X_7} P(X_1, X_2, \ldots, X_8)$. This approach requires storage of the joint *pdf* of all the variables; hence, its space complexity is $O(m^n)$, where m is

the number of distinct values taken by each variable and n is the total number of variables in the DAG. However, there exists an efficient method of computing the *pdf* of X_8 by factoring the joint distribution and performing efficient marginalization of the factors. For example, $P(X_8)$ can be obtained by (2) shown at the bottom of the page.

In (2), the summation over a variable X_i (or marginalizing X_i), $i = 1, \ldots, 7$, represents that the *pdf* is summed over all the possible outcomes of X_i. As a result of marginalizing X_i, the resulting distribution will not be a function of X_i. The order of the marginalizations in (2) is $(X_3, X_1, X_2, X_4, X_5, X_6, X_7)$. This requires the joint *pdf* of at most three variables to be stored after every marginalization step. The method is based on separating the nodes in the DAG into different subsets (also known as cliques) such that the joint *pdf* of the nodes in a subset $\mathcal{C}$ can be computed and the marginal *pdf* of any node in $\mathcal{C}$ can be obtained from these joint distributions. To compute these subsets, the DAG is first converted into a moral graph. The moral graph is then triangulated. A secondary structure (called clique tree [14]) $\mathcal{T}$ is constructed from this triangulated graph and the conditional *pdfs* (CPDs) are assigned to the cliques in $\mathcal{T}$. These probabilities are then propagated within $\mathcal{T}$ to obtain the *pdf* of the primary outputs.

Another method to compute the *pdf* of the arrival times at the outputs by propagating the arrival time *pdfs* through the circuit was proposed in [6]. The authors obtain the set of dependence nodes (nodes that are a source of reconvergent fan-outs) and then enumerate all the possible outcomes of the arrival times over the set of these dependence nodes to compute the *pdf* of the arrival times at the outputs. From Fig. 1, X_2 is a dependent node. Hence, for each arrival time at X_2, a conditional probability distribution for X_8 (conditioned on the arrival time at X_2) is computed. In the end, these different CPDs are weighted by the probability of their respective arrival times at X_2 to compute the probability distribution of X_8. In comparison, the BN-based approach marginalizes out X_2 much before the probability distributions of the arrival times at X_5 or X_6 are computed.

A. Graph Moralization and Triangulation

Let P_X be the parent set of a node X in a DAG. X is called the child of every node in P_X. For example, the parent set of X_8 in the DAG of Fig. 1 is $\{X_7, X_6\}$.

Definition 3.2: A moral graph of a DAG G is an undirected graph with nodes and edges of G such that all the nodes in G having a common child are connected to each other. The process of constructing a moral graph is called moralization.

Moralization is done by adding undirected edges between the parents of each node in the DAG and removing the directionality of all the edges in the DAG. Fig. 2 shows the graph of Fig. 1 after moralization. Given the joint *pdf* of the parents $P_X = \{X_1, X_2, \ldots, X_k\}$ of any node X, the joint distribution of X and P_X can be determined using the Bayes theorem [15]

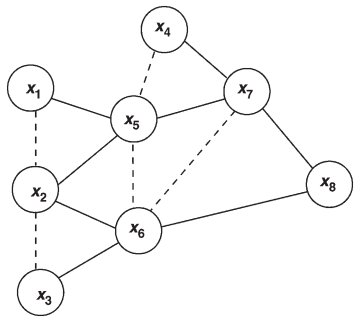

Fig. 2. Moralization of the DAG shown in Fig. 1.

$$P(X, X_1, \ldots, X_k) = P\left(X/X_1, \ldots, X_k\right) P(X_1, \ldots, X_k). \tag{3}$$

Moralization ensures that for every node, there exists a set that contains both the node and its parents.

Definition 3.3: A graph is called complete if there is an edge between every pair of vertices in the graph. A clique is a maximal complete subgraph.

Thus, for every node X in a moralized graph, there is at least one clique containing X and its parents P_X.

Definition 3.4 [16]: An undirected graph $G = (V, E)$ is said to be triangulated (or chordal) if every cycle of length four or more has at least one chord, i.e., an edge joining two nonconsecutive vertices along that cycle.

To obtain an efficient ordering for performing the marginalizations as shown in (2), a clique tree needs to be created from the moral graph.

Definition 3.5: A clique tree $\mathcal{T}$ obtained from a graph G is a tree whose vertices are the cliques of G, such that for any vertex v of G, if we remove from $\mathcal{T}$ all cliques (vertices of $\mathcal{T}$) not containing v, the remaining subtree stays connected. In other words, any two cliques containing v are either adjacent in $\mathcal{T}$ or connected by a path made entirely of cliques that contain v.

From Theorem 3.1, a clique tree exists for a graph only if the graph is triangulated. Hence, the moral graph is triangulated to remove any chordless cycle of length greater than 3.

Theorem 3.1 [16]: Let G be an undirected graph $G = (V, E)$. Then there exists a clique tree corresponding to G if and only if G is triangulated.

The problem of obtaining an optimal triangulation to minimize the maximum clique size in a graph is NP-hard [13]. However, given a moral graph, Algorithm 1 [17] gives an

$$\sum_{X_7, X_6} P(X_8/X_7, X_6) \left(\sum_{X_5, X_4} P(X_7/X_5, X_4) P(X_4) \left(\sum_{X_2} P(X_2) \left(\sum_{X_1} P(X_5/X_2, X_1) P(X_1) \left(\sum_{X_3} P(X_6/X_3, X_2) P(X_3) \right) \right) \right) \right) \tag{2}$$

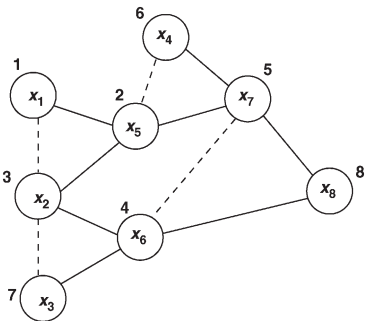

Fig. 3. Result of step 3 in Algorithm 1.

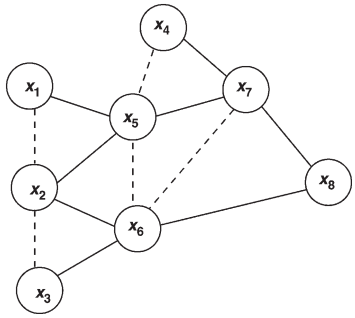

Fig. 4. Triangulated DAG.

efficient two-step algorithm for both testing the chordality of a graph and triangulating it.

Algorithm 1: Graph triangulation algorithm

1) **Input:** a moralized graph.
2) **Output:** a triangulated graph.
3) (Maximum Cardinality Search) Compute an ordering for the nodes: number vertices from 1 to $|\mathbf{V}|$, in increasing order, always assigning the next number to the vertex having the largest set of previously numbered neighbors (breaking ties arbitrarily).
4) From $n = |\mathbf{V}|$ to $n = 1$, recursively fill in edges between any two nonadjacent parents of n, i.e., neighbors of n having lower ranks than n (including neighbors linked to n in previous steps). If no edges are added, the graph is triangulated; otherwise, the new filled graph is triangulated.

The complexity of maximum cardinality search (Step 3) in Algorithm 1 is $O(n + e)$, where n is the number of nodes and e is the number of edges in the graph. Fig. 3 shows the graph with its nodes numbered as a result of maximum cardinality search. After obtaining the ordering of the nodes, the chords are added to obtain the triangulated graph shown in Fig. 4.

From a triangulated graph, the cliques of the graph can be obtained using Algorithm 2 [13] in $O(n + e)$ time. The subgraph on nodes X_6, X_7, and X_8 in the triangulated DAG in Fig. 4 forms a clique. Table I shows all the cliques of this triangulated graph. Thus, the original circuit graph has now been partitioned into different cliques so that the distribution of a node X can be obtained by computing the joint distribution of nodes in a clique $\mathcal{C}$ such that $X \in \mathcal{C}$.

TABLE I
CLIQUES OF THE TRIANGULATED GRAPH SHOWN IN FIG. 4

Clique	Vertices	Rank	Clique	Vertices	Rank
C_1	$\{X_1, X_2, X_5\}$	3	C_4	$\{X_4, X_5, X_7\}$	6
C_2	$\{X_2, X_3, X_6\}$	7	C_5	$\{X_5, X_6, X_7\}$	5
C_3	$\{X_2, X_5, X_6\}$	4	C_6	$\{X_6, X_7, X_8\}$	8

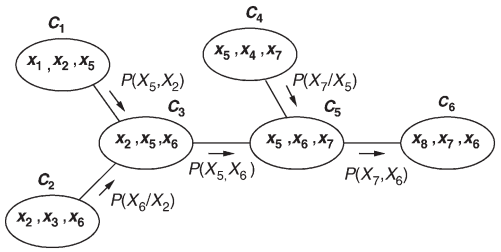

Fig. 5. Clique tree of the moralized graph in Fig. 2.

Algorithm 2: Generating cliques from a triangulated graph

1) **Input:** a triangulated graph $G(V, E)$.
2) **Output:** set of cliques $\{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k\}$ of G. Start with the highest ranked vertex X in G, X and its neighbors form a clique.
3) Remove X from the vertex set $V = V \setminus X$ and remove all the edges incident on X from E. Also, remove all the nodes present only in the neighbor set of X.
4) If $V = \phi$, then we have obtained all the cliques. Otherwise go to step 3.

B. Clique Tree: A Secondary Structure

As a first step towards obtaining the joint *pdf* of the variables in a clique, a clique tree (shown in Fig. 5) is constructed using Algorithm 3 [13], [16], [18] from the triangulated graph. An edge between two cliques represents that there exists at least one common variable between the two cliques. The first step in constructing a clique tree is to order the cliques according to the rank of the highest vertex in each clique. From the method used for obtaining the cliques (Algorithm 2), it can be seen that in every step the highest ranked vertex X is removed along with a clique containing X. Thus, the rank of the cliques obtained in every step will be lower than the rank of the clique obtained in the previous step. This guarantees that there are no two cliques with the same rank. Table I shows the rank of the highest ranked vertex in each clique. It can be seen that the ordering of the cliques in the increasing order of their ranks is C_1, C_3, C_5, C_4, C_2, C_6. The cliques are now connected to each other starting with C_6. Since the next clique is lower in order than C_6 and sharing maximum number of variables with it is C_5, they are joined. Now C_2, which is the next clique in order, is joined with C_3, which is lower in order and shares the maximum number of variables with C_2. The edges are added in this manner until all the cliques have been processed to obtain the clique tree shown in Fig. 5.

Algorithm 3: Assembling a clique tree

1) **Input:** a moralized graph G.
2) **Output:** a clique tree for G.
3) Triangulate the graph G using Algorithm 1 to generate a chordal graph G' (if G is chordal, $G' = G$).

TABLE II
ASSIGNMENT OF PROBABILITY DISTRIBUTIONS TO THE CLIQUES

Clique	Probability Distribution	Clique	Prob. Dist.
C_1	$P(X_5/X_1, X_2), P(X_1), P(X_2)$	C_3	1
C_2	$P(X_6/X_2, X_3), P(X_3)$	C_5	1
C_4	$P(X_7/X_4, X_5), P(X_4)$	C_6	$P(X_8/X_6, X_7)$

4) Identify all cliques in G' using Algorithm 2. Let the number of cliques be k. Since any vertex and its parent set (lower ranked nodes connected to it) form a clique in G', the maximum number of cliques can be $|\mathbf{V}|$.
5) Order the cliques $C_1, C_2, \ldots, C_k$ by rank of the highest vertex in each clique in ascending order.
6) Starting with C_k, form the clique tree by connecting each C_i to a predecessor C_j $(j < i)$ sharing the highest number of vertices with C_i.

C. Assigning Probabilities to Cliques

Once the clique tree is constructed, the CPDs/marginal *pdfs* of every variable X need to be assigned to a clique containing X. Algorithm 4 [14] shows the steps involved in doing this. The input to the algorithm is the set of marginal probability functions associated with the primary inputs and the conditional probability functions of each child conditioned on its parents. If n is the number of nodes in the BN, there are n probability functions (both conditional and marginal). Also, the maximum number of cliques in a graph can be n. Hence, the complexity of assigning the probabilities is $O(n^2)$, where n is the number of nodes in the graph. Table II shows the cliques and the *pdfs* assigned to them. If after assigning all the *pdfs* there exist cliques to which no *pdf* has been assigned, such cliques are assigned 1 to indicate that no *pdf* has been assigned to that clique.

Algorithm 4: Probability assignment to cliques

1) **Input:** set of probability functions $\mathcal{P} = \{P_1, \ldots, P_n\}$.
2) **Output:** set of cliques $\mathcal{C} = \{C_1, \ldots, C_k\}$.
3) Starting with P_1, assign P_i to a clique C_j such that C_j contains all the variables over which P_i is defined (or the domain of P_i), breaking ties arbitrarily.
4) To every clique $C_j \in \mathcal{C}$ that does not have any P_i assigned to it, assign 1.

D. Propagation in a Clique Tree

To obtain the marginal *pdf* of the arrival times at a primary output O_k of a circuit, the joint *pdfs* of the variables in a clique $\mathcal{C}$ need to be computed such that $O_k \in \mathcal{C}$. The clique $\mathcal{C}$ is made the root of the clique tree. The leaf nodes of this rooted clique tree contain at least one variable corresponding to the primary inputs. For example, in the clique tree shown in Fig. 5, C_6 is the root and C_1, C_2, and C_4 are the leaves. It can be seen that every leaf node contains at least one primary input. The joint *pdf* of the variables in the root clique is computed by propagating the *pdfs* of the variables in the leaf nodes to the root.

The joint *pdf* of the variables for each of the leaf nodes (cliques) in the clique tree is calculated by multiplying the *pdfs* assigned to that clique. For example, the joint *pdf* of the variables in C_1 is given by

$$P(X_1, X_2, X_5) = P(X_5/X_1, X_2)P(X_1)P(X_2). \quad (4)$$

Equation (4) is valid under the assumption that the arrival times at the primary inputs are independent of each other. Similarly, the *pdf* of the variables in C_2 can also be obtained.

A clique $\mathcal{C}_Y$ is a child of clique $\mathcal{C}_X$ if $\mathcal{C}_Y$ is adjacent to $\mathcal{C}_X$ and $\mathcal{C}_X$ lies on the path from $\mathcal{C}_Y$ to the root of the clique tree. Once the *pdfs* of all the variables in every child clique $(\mathcal{C}_{Xi})$ of a clique $\mathcal{C}_X$ have been computed, they are propagated from the child cliques to the parent clique. However, some of the variables can be marginalized from the joint *pdf* of the child clique $\mathcal{C}_Y$ before propagating it to the parent clique $\mathcal{C}_X$. From Theorem 3.1, if a variable A is present in a child clique $\mathcal{C}_Y$ and is not in the domain of its parent $\mathcal{C}_X$, then A will not be present in any clique on the path from $\mathcal{C}_X$ to the root clique of the tree. Hence, A can be marginalized from the joint *pdf* of the variables in $\mathcal{C}_Y$ before propagating it to $\mathcal{C}_X$. For example, while propagating the *pdf* from C_1 to C_3, the joint *pdf* of X_1, X_2, and X_5 can be summed over X_1 since $\{X_1, X_2, X_5\} \setminus \{X_2, X_5, X_6\} = \{X_1\}$.

The *pdf* of the variables in a parent clique $\mathcal{C}_X$ can then be computed by multiplying the *pdfs* incident on $\mathcal{C}_X$ and the ones assigned to $\mathcal{C}_X$. Thus, the distribution of the variables in C_3 is obtained as

$$P(X_6, X_5, X_2) = \left(\sum_{X_1} \phi_{C_1}\right)\left(\sum_{X_3} \phi_{C_2}\right) \quad (1)$$
$$= P(X_6/X_2)P(X_5, X_2) \quad (5)$$

where ϕ_{C_1} is $P(X_5, X_2, X_1)$ and ϕ_{C_2} is $P(X_6, X_3/X_2)$. Equation (5) is obtained from the independence of the random variables X_5 and X_6 given X_2. This shows that the *pdf* of the variables in C_3 can be obtained from that of C_2 and C_1. Fig. 5 shows how all the *pdfs* are propagated in the clique tree. Following the same procedure, the joint distribution of the variables in C_6 is computed from which the marginal distribution of X_8 can be computed.

E. More on Complexity

1) Space Complexity: The space complexity of the entire procedure is $O(m^{|C|})$, where $|C|$ is the number of variables in the largest clique and m is the maximum number of distinct values taken by a variable. The maximum clique size present in a BN is dependent on the amount of reconvergence in the network as well as the maximum fan-in in the circuit. Since the maximum fan-in in a circuit is bounded (typically 10–15), in practice the clique sizes will be much smaller than the circuit size.

Algorithm 1 is an efficient heuristic for triangulation to produce small clique sizes. The fourth column in Table III shows the maximum clique sizes for the International Symposium on Circuits and Systems (ISCAS) benchmark circuits. This confirms the assumption that the clique size grows much slower than the circuit size. Thus, BNs are efficient tools for computing

TABLE III
MAXIMUM CLIQUE SIZE IN BENCHMARK CIRCUITS

Circuit	Nodes	Edges	Max. Clique size	No. of Cliques
C17	12	14	4	9
C432	197	343	40	150
C499	244	440	31	184
C880	444	755	54	326
C1355	588	1096	59	402
C1908	914	1522	76	678
C2670	1427	2214	84	1094
C3540	1720	2958	160	1195
C5315	2486	4508	153	1716
C6288	2449	4831	232	1519
C7552	3720	6250	83	2625

the *pdf* of any variable in the network. As is evident from this example, the complexity is directly related to the maximum clique size of the DAG.

The dependence of the maximum clique size on the amount of reconvergence and the density of the DAG can be seen from Fig. 6. The DAGs G_A and G_B in Fig. 6 have the longest cycle of length 7. However, G_B has more number of nodes with reconvergent fan-ins. From their corresponding triangulated graphs, it can be seen that the maximum clique size for the triangulated graph of G_B is larger than the triangulated graph of G_A. Hence, the complexity of analyzing G_B is higher when compared to G_A.

If there are k levels in the DAG, and each gate can take n distinct delay values, then the number of distinct arrival times at a primary output will be $m = n \cdot k$. Hence, increasing the resolution of the gate delay increases the complexity of computing the *pdf* of the circuit delay.

2) Time Complexity: The time complexity of generating a clique tree $\mathcal{T}$ from a DAG G and assigning the probability distributions to the cliques in $\mathcal{T}$ is $O(n^2)$, where n is the number of nodes in G. Computation of the joint *pdf* of the variables in a clique $\mathcal{C}$ is equivalent to obtaining a k-dimensional matrix (where k is the number of variables in $\mathcal{C}$). Every element of this k-dimensional matrix is equal to the product of the elements of l matrices, where l is the number of neighbors of $\mathcal{C}$. Thus, the complexity of obtaining this k-dimensional matrix is $O(l \cdot m^k)$. The complexity of obtaining the marginal distribution of a variable from the joint *pdf* of k variables is $O(m^k)$. Hence, the time complexity of the entire procedure is $O(n \cdot l \cdot m^{|C|})$, as there can be at most n cliques.

For this work, the BN Toolbox for MATLAB reported in [19] was used. To specify the BN, the CPDs $P(X/X_j, X_{j+1}, \ldots, X_k)$ for each of the nodes (X) in the circuit are constructed as

$$P(X/X_j, \ldots, X_k) = \sum_d P(X, D = d/X_j, \ldots, X_k) = \sum_d P(X/X_j, \ldots, X_k, D) P(D = d). \quad (6)$$

Now, $P(X/X_j, X_{j+1}, \ldots, X_k, D)$ is either 0 or 1 for a particular combination of $X, X_j, \ldots, X_k$ and D. Since the distribution

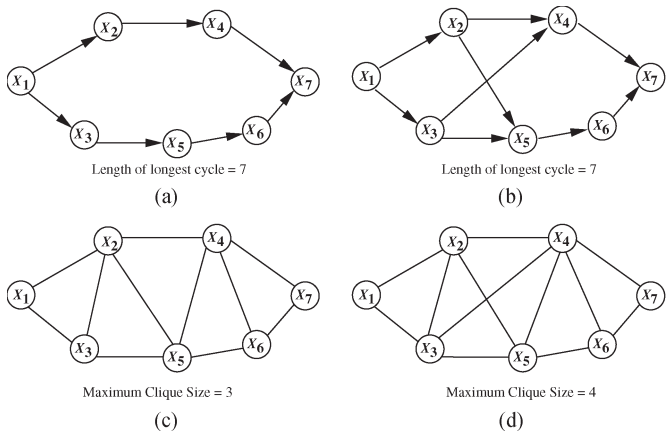

Fig. 6. Dependence of the maximum clique size on the amount of reconvergence and density of a DAG. (a) Original DAG G_A. (b) Original DAG G_B. (c) Triangulated graph obtained from G_A. (d) Triangulated graph obtained from G_B.

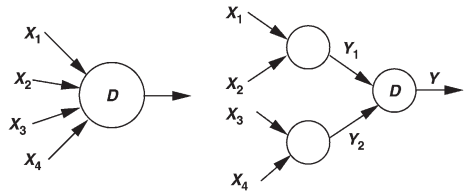

Fig. 7. Reducing the maximum fan-in in the DAG.

of the delay can be assumed to be given, the CPDs for each of the variables can be specified to perform the analysis.

IV. GRAPH TRANSFORMATIONS

The complexity of constructing the exact *pdf* of the circuit delay can be reduced either by implementing efficient triangulation algorithms or by devising new methods to reduce the circuit size. The former will result in only an incremental improvement in speed, whereas reducing the circuit size will lead to tight lower bounds on the *pdf* of the delay. In this section, we present a set of transformations that can be used to reduce the complexity of obtaining the *pdf* of the signal arrival times. The size for storing the CPDs is reduced by performing the reduce fan-in transformation so that the maximum fan-in in the reduced DAG is 2. The next two transformations, switching window reduction (SWR) and inputs reduction, help in eliminating nodes that do not contribute to the critical delay of the circuit. Finally, series reduction transformation is used to combine nodes satisfying a particular topology. We also prove the correctness of these transformations.

A. Fan-ins Reduction

The size of the CPD depends exponentially on the fan-in of a node. Hence, if a node has fan-in k, with input having m distinct values, then the size required to store this distribution is $O(m^{k+1})$ (including one additional dimension for the output of the node). This size is extremely large even for a small value of m. To reduce this size, the node is split into several nodes in the form of a tree as shown in Fig. 7. The delay associated with each of the new nodes is 0, whereas the root node has

delay of the original node associated with it. Equation (7) shows that this transformation results in a graph that has an identical underlying *pdf* as the original graph, i.e.,

$$\begin{aligned} &P\left(\max\{X_1, X_2, X_3, X_4\} + D \le t\right) \\ &\quad = P\left(\max\left\{\max\{X_1, X_2\}, \max\{X_3, X_4\}\right\} + D \le t\right) \\ &\quad = P\left(\max\{Y_1, Y_2\} + D \le t\right). \end{aligned} \tag{7}$$

This transformation is performed to restrict the maximum fan-in of a node in the DAG to 2. Thus, the size of the largest CPD will be $O(m^3)$. In the process, at most $2^{\lceil \log_2 k \rceil}$ new nodes are added, where k is the number of fan-ins. The space required in the original case is m^{k+1}, whereas the space required in the transformed DAG is $(2^{\lceil \log_2 k \rceil})m^3$. Hence, we get a reduction in the space required for $k \ge 3$ and $m \ge 2$. Thus, although the number of nodes increases, the total space required to store the CPD decreases.

B. Reducing Switching Window Size

The switching window of a signal X is an interval $[l_X, u_X]$, where l_X and u_X are the earliest and latest arrival time (EAT and LAT) at X, respectively. In general, a designer is interested in computing the distribution of the arrival times over the entire switching window of the output. However, of greater interest for the purpose of timing yield analysis is distribution close to the LAT. Hence, by removing some events that result in an arrival time close to the EAT of the outputs, a significant number of nodes can be discarded without sacrificing accuracy. The circuit size can be reduced using this idea by propagating a critical time (T^*) from the primary outputs to the primary inputs just as required time is propagated in static timing analysis. A sink node X_{sink} is created and a directed edge is added from each output O_i to X_{sink} such that X_{sink} has a zero delay associated with it.

Algorithm 5 shows the steps in the SWR transformation. The complexity of this algorithm is $O(fn)$, where f is the maximum fan-in of the nodes and n is the total number of nodes in the circuit graph.

Algorithm 5: SWR transformation

1) **Input:** DAG $G = (V, E)$ with the sink node and a required time T^*.
2) **Output:** Reduced DAG $G' = (V', E')$.
3) Label the nodes in topological order. Hence, the label on $X_{\text{sink}} = |V|$.
4) $T^*_{X_{\text{sink}}} := T^*$ and $T_X := l_X$ for all nodes $X \in V$.
5) For all nodes in V, starting with the node X labeled $n = |V|$ in the decreasing order, perform Steps 6 and 7.
6) Assign the required time $T^*_{Y_i}$ to the node Y_i in the fan-in cone of X as $T^*_{Y_i} := \max\{l_X - d_{X_{\min}}, T^*_{Y_i}\}$.
7) If $T^*_X > u_X$ for some node X, remove X from the graph. Also, remove all nodes in the fan-in cone FI_X of X that do not fan-out to any node outside FI_X. Go to Step 5.
8) The resultant graph is G'.

The SWR transformation is shown over a simple graph structure in Fig. 8. The original graph is shown in Fig. 8(a), with

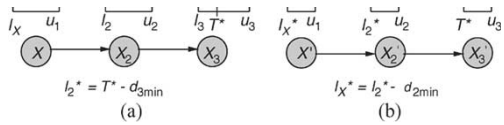

Fig. 8. (a) Original DAG and (b) DAG after SWR transformation.

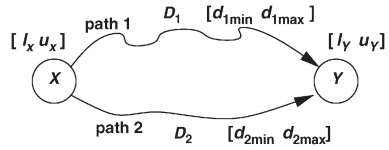

Fig. 9. Dependencies between the fan-ins.

$[l_i, u_i]$ being the switching interval for variable X_i. The delay of each node X_i can take values in the range $[d_{i\min}, d_{i\max}]$. T^* is the required time at the output X_3 and is propagated backward as explained in Algorithm 5.

Definition 4.1 [20]: If X and Y are two random variables with sample spaces S_X and S_Y, $S_Y \subseteq S_X$, then Y is s.l. than X, denoted by $Y \ge_{\text{st}} X$, if $P(Y > t) \ge P(X > t)\ \forall\, t \in S_Y$.

If Y is s.l. than X, then the *pdf* of Y is a lower bound on the *pdf* of X. Hence, the fraction of circuits whose delay is greater than a critical value t (i.e., $P(X > t)$) is never underestimated if X is replaced with an s.l. random variable Y.

In the following, the term reduced DAG means the DAG obtained after performing SWR transformation on the original DAG. 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 $Y_{\text{red}} \ge_{\text{st}} Y$.

Let X' be a primary input in the reduced DAG and X be the corresponding input in the original DAG. Let T^*_X be the required time associated with X as a result of propagating a required time T^* from the sink node to the primary inputs. Then $X' = \max\{X, T^*_X\}$. Since $\{X' \le t\} \equiv \{X \le t, T^*_X \le t\} \subseteq \{X \le t\}$, $P(X' \le t) \le P(X \le t)$ or $P(X' > t) \ge P(X > t)$. Hence, $X' \ge_{\text{st}} X$. Thus, all the variables representing primary inputs in the reduced DAG are s.l. than the corresponding variables in the original DAG. The *pdfs* of the arrival times of the primary inputs in reduced DAG are obtained using

$$P(X' \le t) = \begin{cases} 0, & t < l^*_X \\ P(X \le t), & t \ge l^*_X. \end{cases} \tag{8}$$

Now consider an arbitrary node Y' in the reduced graph and let Y be the corresponding node in the original DAG. After performing the reduce fan-in transformation, either the two inputs to Y' originate from the same node as shown in Fig. 9, or they are independent. Lemma 4.1 shows that in the case similar to shown in Fig. 9, Y' is s.l. than Y.

Lemma 4.1: Let X and Y be two random variables denoting the arrival times of two nodes as shown in Fig. 9 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 corresponding to the delay along two paths with sample space $[d_{1\min}, d_{1\max}]$ and $[d_{2\min}, d_{2\max}]$, respectively. Also, let X' and Y' be two random variables representing the arrival times of the corresponding nodes in the reduced DAG

with sample space $[l_X^*, u_X]$ ($l_X^* \geq l_X$) and $[l_Y^*, u_Y]$ ($l_Y^* \geq l_Y$), respectively, such that

$$X' = \max\{X, l_X^*\}, \quad Y' = X' + \max\{D_1, D_2\}$$

then Y' is a lower bound of Y.

Proof: It was shown earlier that $X' \geq_{\text{st}} X$. It should be noted that the EAT of the node Y' is the sum of the maximum of the minimum delays along D_1 and D_2, and the EAT at node X', that is, $l_Y^* = l_X^* + \max\{d_{1\min}, d_{2\min}\}$. Now the *pdf* of Y' is given by

$$\begin{aligned} &P(Y' \leq t) \\ &\quad = P\left(X' + \max\{D_1, D_2\} \leq t\right) \\ &\quad = \sum_{d_1, d_2} P\left(X' \leq t - \max\{d_1, d_2\}/d_1, d_2\right) P(d_1, d_2) \quad (9) \\ &\quad \leq \sum_{d_1, d_2} P\left(X \leq t - \max\{d_1, d_2\}/d_1, d_2\right) P(d_1, d_2) \\ &\quad = P(Y \leq t). \quad (10) \end{aligned}$$

From (10), we see that $Y' \geq_{\text{st}} Y$. ■

In the proof of Lemma 4.1, there was no condition on the independence of delay of the two paths D_1 and D_2; hence, even if the delays of the two path are dependent due to reconvergence, the transformation is still valid.

It has been shown that the arrival times of the primary inputs in the reduced DAG are s.l. than the arrival times of the primary inputs in the original DAG. It was also shown that if two paths from a variable X' that is s.l. than the corresponding variable X in the original DAG reconverge at some variable Y', then Y' will also be s.l. than the corresponding variable Y in the original DAG. To complete the validity of this transformation, it needs to be shown that the signals at second level of the reduced DAG are also s.l. than the signals at second level in the original DAG.

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 represent a random variable 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' is a lower bound of Y.

Proof: Since Y' represents the arrival time corresponding to Y in the reduced DAG, we have $Y' = \max\{X_1', X_2'\} + D$. Thus, the *pdf* of Y' ($P(Y' \leq t)$) is obtained using

$$\begin{aligned} &P\left(\max\{X_1', X_2'\} + D \leq t\right) \\ &\quad = \sum_d P\left(X_1' \leq t - d, X_2' \leq t - d/D = d\right) P(D = d) \\ &\quad \leq \sum_d P(X_1 \leq t - d) P(X_2 \leq t - d) P(D = d) \quad (11) \end{aligned}$$

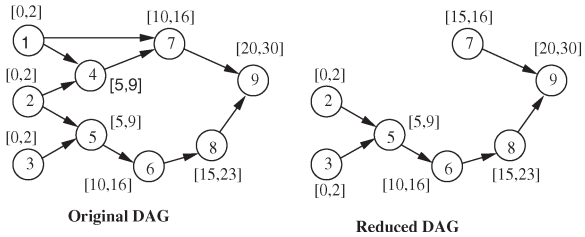

Fig. 10. SWR transformation.

$$\begin{aligned} &= \sum_d P\left(\max\{X_1, X_2\} \leq t - d/D = d\right) P(D = d) \\ &= P(Y \leq t). \quad (12) \end{aligned}$$

Since $X' \geq_{\text{st}} X$, (11) can be obtained. ■

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

Fig. 10 shows the DAG of a circuit and its 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. For simplicity, the delay of each node in the DAG of Fig. 10 has the range [5,7]. If there is an internal node whose fan-ins are removed (e.g., node 7 in Fig. 10), the LAT of that node is assigned a Probability of "1" to ensure that its arrival time is s.l. than the arrival time of the same node in Original DAG.

C. Inputs Reduction

Depending on the relative alignment of the switching windows of the two inputs to a node, further reduction in the graph size can be performed as shown in Theorem 4.2.

Theorem 4.2: If A and B are the fan-ins of a node C such that the LAT of A is less than or equal to the EAT of B, then A can be removed from the fan-in of C without affecting the *pdf* of C.

Proof: The arrival time of C can be written as $T_C = \max\{T_A, T_B\} + D$, where T_A, T_B, and T_C are random variables representing the arrival times of signals A, B, and C. For each possible combination of T_A, T_B, and D, for example, t_A, t_B, and d, $t_A \leq t_B$. We have

$$\begin{aligned} t_C &= \max\{t_A, t_B\} + d \\ &= t_B + d \quad \forall\, t_A,\ t_B,\ t_C, \text{ and } d. \quad (13) \end{aligned}$$

Hence, $T_C = T_B + D$. ■

Thus, the edge from A to C can be removed from the graph. Note that [6] also uses this transformation. In order to remove A from the entire graph, all the fan-outs from A should satisfy

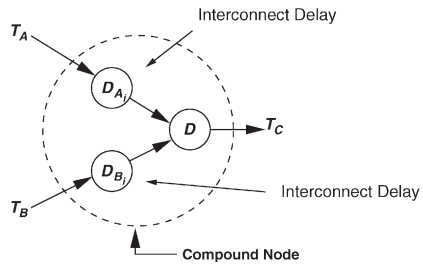

Fig. 11. Compound node for wire delays.

the above condition, i.e., the LAT of A should be less than or equal to the EAT of all other inputs of all the gates in the fan-out of A. Using this transformation, all the nodes in the fan-in cone FI_A of A can be removed if none of the nodes fan-out to a node outside FI_A.

D. Series Reduction

A much widely known form of transformation is also performed, which reduces the complexity of analysis by combining two nodes if the only fan-out of the first node is the second node and the only fan-in of the second node is the first node.

Let the delays of the two nodes be D_1 and D_2, respectively. The delay of the reduced node is $D = D_1 + D_2$. Hence, if the ranges of D_1 and D_2 were $[d_{1\min}, d_{1\max}]$ and $[d_{2\min}, d_{2\max}]$, then range of the resultant node will be $[d_{1\min} + d_{2\min}, d_{1\max} + d_{2\max}]$. Under the assumption of the independence of the gate delays, from general probability theory [15], the probability density of the sum of two independent random variables is obtained by convolving their probability density functions

$$P(D = t) = \int_{-\infty}^{\infty} P(D_1 = t - \tau)P(D_2 = \tau)d\tau. \qquad (14)$$

V. What About Wire Delays?

Even though a significant amount of work has gone in the development of PTA tools, the problem of including wire delays and still being able to perform the analysis efficiently has not been discussed in detail. A naive way to include the wire delays in the analysis is to insert a node on each of the edges of the DAG; however, this procedure increases the DAG size by the number of edges in the DAG. This will result in a DAG size that is significantly larger than the size of the original DAG. In this section, we present an efficient way of incorporating wire delays without increasing the number of nodes in the DAG along with only a minor increase in memory space.

We assume that the delay of an interconnect is independent of the delay of the gate driving it as well as the delay of the gate driven by this interconnect. We first insert dummy nodes corresponding to the interconnect delays on each of the edges as shown in Fig. 11. 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. Our assumption regarding the independence of gate and interconnect delays implies that the 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 *pdf* of T_C conditioned on T_A and T_B, $P(T_C = t/t_A, t_B)$ can be computed as

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

From the independence of D_{A_i} and D_{B_i}, $P_{D_{A_i},D_{B_i}}(d_{A_i}, d_{B_i}) = P_{D_{A_i}}(d_{A_i})P_{D_{B_i}}(d_{B_i})$. Now the first term in the summation is either "0" or "1" depending on whether $\max\{t_A + d_{A_i}, t_B + d_{B_i}\}$ is equal to $t - d$ or not. For a given combination of t_A, t_B, d, d_{A_i}, and d_{B_i} being "1" if it is and "0" otherwise. 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.

TABLE IV
Reduction in the Circuit Size on Performing Transformations

Circuit	Nodes Remaining After		% red.	Maximum Clique Size	Cliques
	Switching Window	Series red.			
C17	8	8	33.33%	3	4
C432	164	93	52.79%	12	58
C499	253	189	22.54%	8	121
C880	129	58	86.91%	6	40
C1355	497	449	23.64%	9	323
C1908	486	344	62.36%	17	280
C2670	463	372	73.93%	11	306
C3540	339	228	86.74%	8	175
C5315	391	256	89.7%	6	181
C6288	1249	1217	50%	23	944
C7552	668	472	87%	7	337

VI. Results

We performed our analysis on ISCAS 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 fan-ins/fan-outs. We ran our simulations on a shared Sun 280r server having two Sparc III processors 900 MHz and 4 GB RAM.

The reduce fan-in and SWR transformations were implemented in PERL and the resulting DAG was given as an input to the MATLAB program. The inputs reduction and series reduction transformations and the remaining procedures were implemented in MATLAB. The BN-based analysis was done using BN Toolbox [19] in MATLAB.

Table IV shows the reduction in gate sizes obtained after performing switching window and series reduction transformations. We obtained as much as 89% reduction in the circuit

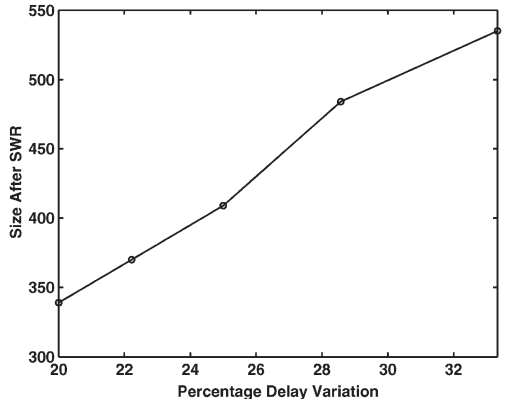

Fig. 12. Circuit size for C3540 after SWR transformation versus variation in gate delay.

TABLE V
COMPARISON OF BN-BASED APPROACH AND MONTE CARLO SIMULATIONS

Circuit	Means (nanosecond)		Standard Deviation (nanosecond)		$\mu + 3\sigma$		Difference (percent)	Runtime (second)	
	MC	BN	MC	BN	MC	BN	$\mu + 3\sigma$	MC	BN
C17	2.02	2.02	0.126	0.126	2.38	2.38	0	7.73	0.61
C432	18.73	18.99	0.241	0.161	19.45	19.48	0.13	209	30
C499	12.76	12.87	0.147	0.117	13.21	13.22	0.06	273	60
C880	25.59	25.76	0.298	0.24	26.47	26.48	0.06	442	10
C1355	25.25	25.83	0.19	0.11	25.81	26.18	1.41	670	510
C1908	25.11	25.93	0.36	0.27	26.22	26.75	1.87	1047	518
C2670	33.74	34.32	0.34	0.27	34.70	35.13	1.08	1487	157
C3540	48.71	50.49	0.38	0.26	49.86	51.28	2.77	1963	570
C5315	29.97	30.73	0.399	0.338	31.14	31.72	1.83	3239	240
C7552	26.97	27.25	0.34	0.29	28.02	28.12	0.36	4014	717

sizes and the average reduction obtained was 61%. The table also shows the maximum clique size as well as the number of cliques in the reduced circuit. From the column of maximum clique size, it can be seen that there is a considerable reduction in the maximum clique size compared to Table III and so in the complexity of the problem.

The percentage variation in the delay $((d_{\max} - d_{\min})/d_{\text{mean}})$ in the analysis was in the range from 20% to 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. The amount of reduction decreases as the percentage variation in the delay increases. This can be attributed to the fact that as the variation increases, more and more paths start becoming critical. For example, in the extreme case of no variations, there might be only one critical path, but in presence of variations there are typically more than one critical path. The effect of the gate delay variation on the amount of reduction in the circuit size for circuit C3540 is shown in Fig. 12. The figures show that the reduced circuit size increases with an increase in the variation in the gate delays.

The exact distribution for C17 was obtained. Because of the large number of reconvergences present in other circuits, only their bounds could be obtained. In Table V, the means and the standard deviation of the output of the circuit obtained from our analysis are compared with Monte Carlo simulations. 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.

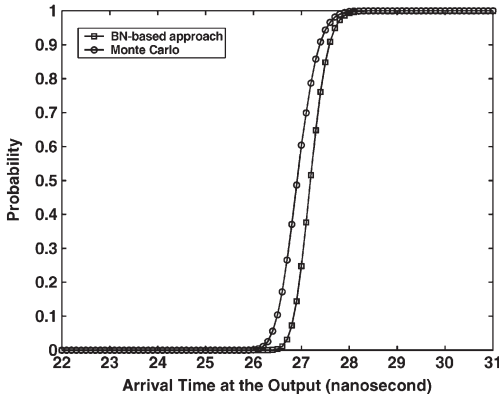

Fig. 13. Probability distribution of the delay of circuit C7552.

TABLE VI
RUNTIME–ACCURACY TRADEOFF FOR C7552

Critical Time (nanosecond)	$\mu + 3\sigma$(nanosecond)		Difference (percent)	Runtime (second)
	MC	BN		
22.00	28.02	28.13	0.39	717.00
22.50	28.02	28.33	1.09	546.06
23.00	28.02	28.71	2.40	391.00

TABLE VII
COMPARISON OF α PERCENTILES FOR C7552 AND C5315

α	C7552			C5315		
	MC (nanosecond)	BN (nanosecond)	Difference (percent)	MC (nanosecond)	BN (nanosecond)	Difference (percent)
10	26.49	26.84	1.291	29.43	30.27	2.77
20	26.63	26.95	1.202	29.60	30.40	2.64
30	26.73	27.04	1.139	29.71	30.50	2.58
40	26.83	27.12	1.076	29.82	30.59	2.50
50	26.91	27.19	1.020	29.92	30.67	2.45
60	27.00	27.26	0.980	30.01	30.75	2.41
70	27.10	27.35	0.896	30.12	30.85	2.35
80	27.22	27.44	0.824	30.25	30.96	2.29
90	27.38	27.59	0.731	30.43	31.12	2.21

The probability distribution of the delay of the circuit C7552 is compared with Monte Carlo simulations in Fig. 13. We see that the predicted probability is always less than the simulated probability. Hence, we never underestimate the fraction of circuits whose delay is greater than a given critical time.

A. Runtime–Accuracy Tradeoff

The runtime of this approach can be reduced at the cost of the quality of the bounds. Table VI shows an example of such a tradeoff on C7552. The different rows correspond to the $\mu + 3\sigma$ values of the circuit delay for the different values of the critical time assigned during the SWR transformation. The switching window of the output is [22, 31] ns. The runtime on assigning the critical time of 22, 22.5, and 23 ns is shown in Table VI. We see that we can reduce the runtime considerably (by around 45%) while incurring only a small penalty in the accuracy.

To compare the tightness of the distributions obtained using the Monte Carlo method and our approach using BNs, the percentiles of the arrival time were computed. The values of the α percentiles for $\alpha = 10, 20, \ldots, 90$ are shown in Table VII for C7552 and C5315. It can be seen that the BN-based technique accurately estimates the α percentiles with the maximum percentage difference for the two cases being less than 3%.

Since the computations have been performed in MATLAB, the runtime of our analysis is much slower than compared to a BN package implemented C. Hence, significant speedups can be obtained by implementing the code in C.

VII. Conclusion

A new methodology for performing probabilistic timing analysis (PTA) of circuits based on Bayesian networks (BNs) was introduced. The problem of finding the exact probability distribution function (*pdf*) of the arrival time of a signal in the circuit was shown to be exponential in the maximum clique size of a graph derived from the circuit. Various analytical results were presented using which different graph transformations can be performed to reduce the circuit size without having significant effect on the accuracy. A method for incorporating wire delays in the analysis was proposed without significant increase in the complexity. The transformations can result in as much as 89% reduction in the circuit size with the average reduction being 61%. 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.

Acknowledgment

The authors would like to thank S. Sapatnekar and the anonymous reviewers who helped in improving the quality of the manuscript.

References

[1] C. Visweswariah, "Death, taxes and failing chips," in *IEEE Design Automation Conf.*, Anaheim, CA, 2003, pp. 343–347.
[2] K. Keutzer and M. Orshansky, "From blind certainty to informed uncertainty," in *Workshop TAU*, Monterey, CA, Dec. 2002, pp. 37–41.
[3] A. Nadas, "Probabilistic pert," *IBM J. Res. Develop.*, vol. 23, no. 3, pp. 339–347, May 1979.
[4] H.-F. Jyu, S. Malik, S. Devdas, and K. Keutzer, "Statistical timing analysis of combinational logic circuits," *IEEE Trans. Very Large Scale (VLSI) Integr. Syst.*, vol. 1, no. 2, pp. 126–137, Jun. 1993.
[5] J.-J. Liou, A. Krstic, L.-C. Wang, and K.-T. Cheng, "False-path-aware statistical timing analysis and efficient path selection for delay testing and timing validation," in *IEEE Design Automation Conf.*, New Orleans, LA, 2002, pp. 566–569.
[6] A. Agarwal, D. Blaauw, V. Zolotov, and S. Vrudhula, "Statistical timing analysis using bounds and selective enumeration," in *Workshop TAU*, Monterey, CA, Dec. 2002, pp. 16–21.
[7] J. A. G. Jess, K. Kalafala, S. R. Naidu, R. H. J. M. Otten, and C. Visweswariah, "Statistical timing for parameteric yield prediction of digital integrated circuits," in *Design Automation Conf.*, Anaheim, CA, 2003, pp. 932–937.
[8] M. Orshansky and K. Keutzer, "A general probabilistic framework for worst case timing analysis," in *Proc. Design Automation Conf. (DAC)*, New Orleans, LA, 2002, pp. 556–561.
[9] A. Agarwal, D. Blaauw, and V. Zolotov, "Statistical timing analysis for intra-die process variations with spatial correlations," in *IEEE Int. Conf. Computer-Aided Design (ICCAD)*, San Jose, CA, 2003, pp. 900–907.
[10] H. Chang and S. S. Sapatnekar, "Statistical timing analysis considering spatial correlations," in *IEEE Int. Conf. Computer-Aided Design (ICCAD)*, San Jose, CA, 2003, pp. 621–625.
[11] S. Bhanja and N. Ranganathan, "Switching activity estimation of VLSI circuits using Bayesian networks," *IEEE Trans. Very Large Scale (VLSI) Integr. Syst.*, vol. 11, no. 4, pp. 558–567, Aug. 2003.
[12] S. Bhardwaj, S. Vrudhula, and D. Blaauw, "τAU: Timing analysis under uncertainty," in *IEEE Int. Conf. Computer-Aided Design (ICCAD)*, San Jose, CA, Nov. 2003, pp. 615–620.
[13] F. V. Jensen, *Bayesian Networks and Decision Graphs*. New York: Springer-Verlag, 2000.
[14] C. Huang and A. Darwiche, "Inference in belief networks: A procedural guide," *Int. J. Approx. Reason.*, vol. 15, no. 3, pp. 225–263, 1994.
[15] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*. New York: McGraw-Hill, 1965.
[16] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, CA: Morgan Kaufmann, 1988.
[17] R. E. Tarjan and M. Yannakakis, "Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs and selectively reduce acyclic hypergraphs," *SIAM J. Comput.*, vol. 13, no. 3, pp. 566–579, 1984.
[18] R. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter, *Probabilistic Networks and Expert Systems*. New York: Springer-Verlag, 1999.
[19] K. Murphy, BN Toolbox. [Online]. Available: http://www.cs.ubc.ca/~murphyk/Software/BNT/bnt.html
[20] S. M. Ross, *Stochastic Processes*. New York: Wiley, 1996.

Sarvesh Bhardwaj (S'05) received the B.Tech. degree in electrical engineering from the Indian Institute of Technology, Delhi, India, in 2000, the M.S. degree in electrical and computer engineering from the University of Arizona, Tucson, in 2003, and is currently working toward the Ph.D. degree in the Electrical Engineering Department, Arizona State University, Tempe.

From 2000 to 2001, he was with MindTree Consulting, Bangalore, India. Since August 2001, he has been working with Prof. S. Vrudhula as a Research Assistant at the Center for Low Power Electronics. His research interests include statistical analysis and optimization of integrated circuits in the presence of process variations, logic synthesis, and design for manufacturability.

Sarma Vrudhula (M'85–SM'01) received the B.Math. (Honors) degree from the University of Waterloo, Waterloo, ON, Canada, in 1976, and the M.S. and Ph.D. degrees in electrical engineering from the University of Southern California, Los Angeles, in 1980 and 1985, respectively.

From 1992 to 2004, he was a Professor in the Electrical and Computer Engineering (ECE) Department, University of Arizona, Tucson. From 2000 to 2001, he was a Visiting Researcher at the Advanced Design Tools Group in Motorola, Austin. He is currently the Consortium for Embedded Systems (CES) Chair Professor in the Department of Computer Science and Engineering, Arizona State University, Tempe, and the Director of the National Science Foundation (NSF) Center for Low Power Electronics. His research interests are in design automation and very large scale integration (VLSI) computer-aided design (CAD). These include optimization problems that arise in chip layout, digital systems testing, logic synthesis and verification, energy-efficient system design, and field-programmable digital and analog arrays. His teaching experience includes both undergraduate and graduate courses in digital systems design and testing, VLSI design, CAD algorithms for VLSI, advanced synthesis and verification methods, computer architecture, and discrete mathematics.

Dr. Vrudhula has served on the technical program committees of many national and international conferences on VLSI CAD, on government review panels, and as an Associate Editor for the IEEE Transactions on Very Large Scale Integrated (VLSI) Systems.

David Blaauw (M'01) received the B.S. degree in physics and computer science from Duke University, Durham, NC, in 1986, the M.S. degree in computer science from the University of Illinois, Urbana, in 1988, and the Ph.D. degree in computer science from the University of Illinois, Urbana, in 1991.

From 1993 to August 2001, he worked for Motorola, Inc., Austin, TX, where he was the Manager of the High Performance Design Technology Group. Since August 2001, he has been an Associate Professor at the University of Michigan, Ann Arbor. His work has focussed on very large scale integration (VLSI) design and computer-aided design (CAD) with particular emphasis on circuit design and optimization for high performance and low power microprocessors.