Switching Activity Estimation of VLSI Circuits using Bayesian Networks

Sanjukta Bhanja and N. Ranganathan, Fellow of IEEE

Abstract— **Switching activity estimation is an important aspect of power estimation at circuit level. Switching activity in a node is temporally correlated with its previous value and is spatially correlated with other nodes in the circuit. It is important to capture the effects of such correlations while estimating the switching activity of a circuit. In this paper, we propose a new switching probability model for combinational circuits that uses a Logic-Induced-Directed-Acyclic-Graph (LIDAG) and prove that such a graph corresponds to a Bayesian Network which is guaranteed to map all the dependencies inherent in the circuit. Bayesian Networks (BN) can be used to effectively model complex conditional dependencies over a set of random variables. The BN inference schemes serve as a computational mechanism that transforms the LIDAG into a junction tree of cliques to allow for probability propagation by local message passing. The proposed approach is accurate and fast. Switching activity estimation of ISCAS and MCNC circuits with random and biased input streams yield high accuracy (average mean error = 0.002) and low computational time (average elapsed time including CPU, memory access and I/O time for the benchmark circuits = 3.93 seconds).**

I. Introduction

Power consumption in a CMOS circuit can be classified into three categories: (i) static leakage power, (ii) short-circuit power and (iii) dynamic power. In a well-designed circuit, the total power dissipation is dominated by the dynamic power consumption of the nodes which arises due to the charging and discharging of the parasitic capacitance during switching [20]. The average power consumption at a gate is given by $P(x) = 0.5V_{dd}^2 f_{clk} C_{load} sw(x)$ where V_{dd} is the supply voltage, f_{clk} is the clock frequency, C_{load} is load capacitance and $sw(x)$ is the average switching activity of the output node x.

It is proven that for zero-delay model of a combinational circuit, only first order temporal correlation is exhibited [15] as the signal possesses Markov property. We are hence interested to capture al higher order spatial correlations and first order temporal correlation to encapsulate all the spatio-temporal dependencies in the circuit. It is our observation that the Bayesian Network is a powerful tool to model switching activity preserving the various dependencies in a circuit. Further, elegant inference mechanisms exist for Bayesian network computation that make the estimation time-efficient and thus, usable for large circuits.

In this work, we model the switching in a combinational circuit using a probabilistic Bayesian Network [2], [3], which allows us to capture both the temporal and spatial dependencies in a comprehensive manner. Bayesian networks are directed acyclic graph (DAG) representations whose nodes represent random variables and the links denote direct dependencies, which are quantified by conditional probabilities of a node given the states of its parents. This DAG structure essentially models the joint probability function over the set of random variables under consideration in a compact manner. The attractive feature of this graphical representation of the joint probability function is that not only does it make conditional dependency relationships among the nodes explicit it also serves as a computational mechanism for efficient probabilistic updating. Bayesian networks have traditionally been used in artificial intelligence and image analysis. Their use in power estimation is new.

The core idea is to express the switching activity of a circuit as a joint probability function which can be mapped one-to-one onto a Bayesian Network which preserves the dependency model of the probability function. We first construct a Logic-Induced-Directed-Acyclic-Graph (LIDAG) based on the logical structure of the circuit. Each signal in the circuit is a random variable in the LIDAG that can have four possible states indicating the transitions from $0 \rightarrow 0, 0 \rightarrow 1, 1 \rightarrow 0, 1 \rightarrow 1$. Directed edges are drawn from the random variables representing switching of the inputs to the random variable for switching at the output of each gate. We prove that the LIDAG, thus obtained, is a Bayesian Network which is the minimal representation that captures all the independency relationships in the circuit.

The salient features of the Bayesian Network (BN) model for switching activity estimation are as follows.

1. *It is able to produce globally correct estimates of switching activity accounting for spatial correlation and by using only local message passing for probability updates which makes the estimation process quick and efficient.*
2. *The computations are parallelizable and multi-threadable.*
3. *It can accommodate input correlation, temporal,*

S. Bhanja and N. Ranganathan are with University of South Florida, Tampa, FL 33620, USA. E-mail: bhanja@eng.usf.edu, ranganat@csee.usf.edu

and spatial correlations efficiently.
4. *Bayesian networks can accept evidence for any node and propagate the evidence in any direction.* For example, it can accept switching activity statistics of an *output* node and update the switching probabilities at all the other nodes. Thus, the network can give switching probability estimates for each node, conditioned on information about switching activity at the input or output nodes, or even at any internal node.
5. *After a compilation process that converts the Bayesian network into a junction tree of cliques, further computation time is small.* Thus, repeated computation of switching activity of the circuit with different input statistics does not require much time.
6. *Finally, a Bayesian network in addition to pair-wise correlations [6] also models conditional independence over a set of variables, which allows it to cover a large class of probabilistic dependencies.*

The organization of the paper is as follows: The relevant previous works are discussed in Section II. The proposed Bayesian Network model is presented in Sections III and IV. Section V describes the computational strategies. We report experimental results in Section VI and provide conclusions and directions for further work in Section VII.

II. Related Work

Non-simulative power estimation techniques have been proposed in the past which have reasonable accuracy and extremely fast estimation procedures. They can be purely non-simulative [8], [4], [7] or they can be statistically simulative[17]. In statistical simulation, the estimated power is highly input sensitive. Although, Monte Carlo simulation technique can use input selection efficiently [17], these techniques have problems when the estimation process has to be performed under correlated input streams.

The non-simulative statistical techniques use knowledge about input statistics to estimate the switching activity of internal nodes. In some of the pioneering works around this idea, Najm *et al.* [14] estimated the mean and variance of current using probability waveforms. In [8], the concept of transition density is introduced and is propagated throughout the circuit by Boolean difference algorithm. However, these methods have problems in handling correlation between nodes and hence, the estimates are inaccurate when the nodes are highly correlated. An accurate way of switching activity estimation is proposed in [7] which has a high space requirement. Tagged probability simulation is proposed in [10], which is based on the local OBDD propagation. The signal correlations are captured by using local OBDDs. However, spatio-temporal correlation between the signals is not discussed. Kapoor [12] has modeled structural dependencies and Schneider *et al.* [15] used one-lag Markov model to capture temporal dependence and Tsui *et al.* [13] modeled spatial correlation. Pair-wise correlation between circuit lines were first proposed by Ercolani *et al.* in [9]. Marculescu *et al.* in [19], studied temporal, spatial and spatio-temporal dependencies by capturing pair-wise correlations. In a later effort to capture higher order correlation approximately, Marculescu *et al.* in [6] handled higher order correlation as a composition of pair-wise correlations. Schneider *et al.* [16] proposed another an approximate technique to model higher order spatial correlations.

The theoretical contribution of our work is that the joint probability function of a set of random variables is exactly mapped capturing *higher order correlations between the signals accurately* using Bayesian Network model. Earlier efforts either treat the distribution as a composition of pair-wise correlated signals between all signals [6], [9] or use an approximate solution for capturing spatial correlation [16]. Moreover, the Bayesian Network models *conditional independence of a subset of signals* unlike in [6]. Also, in contrast to [6], the propagation mechanism *does not assume any signal isotropy* of conditional independence. Results show that the technique has high accuracy with low execution times, for combinatorial benchmark circuits.

III. Proposed Bayesian Network Model

In this section, we first prove that the modeling of switching activity of a combinational circuit is an exact model for the underlying joint probability distribution of the whole circuit. Hence, any correlation that is present in the joint probability distribution is captured in such modeling. The choice of states that each random switching variable can have, captures temporal correlation completely. The way in which the conditional probabilities are provided to the boundaries of the Bayesian Network is discussed in Section IV.

First, we present mathematical notions that have been proposed to capture dependencies amongst random variables. Second, we list the conditions under which a Bayesian network can be constructed to capture these dependencies. Based on these foundations, we prove that all the dependencies amongst the switching variables in a combinational circuit can be captured using a DAG (directed acyclic graph) structured Bayesian network that is derived from the underlying circuit structure.

To formalize the concept of dependencies, we first present the concept of conditional independence [1]. The following discussion is fashioned after [3]. We begin with the definition of *conditional* independence among three sets of random variables.

[1]This concept was introduced to the power estimation community by Marculescu *et al.* [6] in the context of switching estimation.

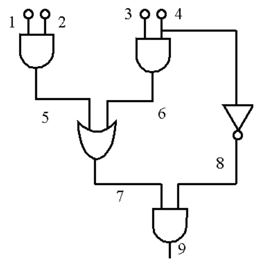

Fig. 1. A small combinational circuit.

Definition 1: Let U=$\{U_1, U_2, \cdots U_n\}$ be a finite set of variables that can assume discrete values. Let $P(.)$ be the joint probability function over the variables in U, and let X, Y and Z be any three subsets of U. X, Y and Z may or may not be disjoint. X and Y are said to be *conditionally independent* given Z if

$$P(x|y,z) = P(x|z) \text{ whenever } P(y,z) > 0 \qquad (1)$$

Following Pearl [3], this conditional independence amongst X, Y, and Z is denoted as $I(X,Z,Y)$ in which X and Y are said to be *conditionally independent* given Z. Conditional independence implies that knowledge of Z makes X and Y independent of each other. In Figure 1 for example, let us denote the switching activity at line i by random variable X_i. U is defined as a set=$\{X_1, \cdots X_9\}$. Switching in a combinational circuit follows directed Markov property since the output of a gate is dependent only on its inputs. Thus the random variable X_9 is completely independent of X_4 given $\{X_6, X_8\}$. Hence, I$(X_9, \{X_6, X_8\}, X_4)$ is one of the many independencies that are present in the circuit.

A dependency model, M, of a domain should capture all these triplets namely (X,Z,Y) conditional independencies amongst the variables in that domain. The joint probability density function is one such dependency model. The properties involving the notion of independence are axiomatized by the following theorem.

Theorem 1: Let X, Y and Z be three distinct subsets of U. If $I(X,Z,Y)$ stands for the relation "X is independent of Y given Z" in some probabilistic model P, then I must satisfy the following four independent conditions:

$$I(X,Z,Y) \Rightarrow I(Y,Z,X)(\text{symmetry}) \qquad (2)$$

$$I(X,Z,Y \cup W) \Rightarrow I(X,Z,Y) \& I(X,Z,W)$$
$$(\text{decomposition}) \qquad (3)$$

$$I(X,Z,Y \cup W) \Rightarrow I(X,Z \cup W,Y)(\text{weak union}) \qquad (4)$$

$$I(X,Z,Y) \& I(X,Z \cup Y,W) \Rightarrow I(X,Z,Y \cup W)$$
$$(\text{contraction}) \qquad (5)$$

Proof: For proof, see [2].

Next, we introduce the concept of *d-separation* of variables in a directed acyclic graph structure (DAG), which is the underlying structure of a Bayesian network. This notion of *d-separation* is then related to the notion of independence amongst triple subsets of a domain.

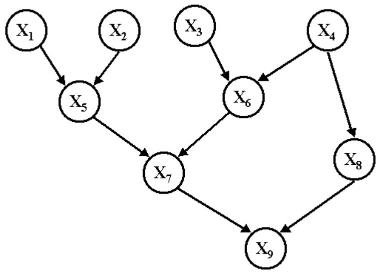

Fig. 2. Bayesian Network corresponding to the circuit in Figure 1.

Definition 2: If X, Y and Z are three distinct node subsets in a DAG D, then X is said to be *d-separated* from Y by Z, $< X|Z|Y >$, if there is no path between any node in X and any node in Y along which the following two conditions hold: (1) every node on the path with converging arrows is in Z or has a descendent in Z and (2) every other node is outside Z. If there exist such a path where the following two conditions hold, the path is called an active path.

In the example DAG in Figure 2, let $X = \{X_6\}$, $Y = \{X_8\}$ and $Z = \{X_9\}$. Path $X_6 \leftarrow X_4 \rightarrow X_8$ is active by condition 2 and the path $X_6 \rightarrow X_7 \rightarrow X_9 \leftarrow X_8$ is active by both condition 1 and 2. Hence X_6 is not d-separated from X_8 by X_9. It is worth mentioning that if any one path is active, even though the other paths are blocked, the nodes are not d-separated. In the same example, X_7 is d-separated from X_2 by X_5 since the only path $X_2 \rightarrow X_5 \rightarrow X_7$ is blocked.

Definition 3: A DAG D is said to be an I-map of a dependency model M if every *d-separation condition* displayed in D corresponds to a valid conditional independence relationship in M, i.e., if for every three disjoint sets of vertices X, Y and Z, we have, $< X|Z|Y > \Rightarrow I(X,Z,Y)$. In Figure 2, for example $< X_7|X_5|X_1 >$ implies the independence relation $I(X_7, X_5, X_1)$ in the dependency model M formed by the random variables depicting switching activity at a line in the combinational circuit in Figure 1.

Definition 4: A DAG is a *minimal* I-map of M if none of its edges can be deleted without destroying its dependency model M.

Definition 5: Given a probability function P on a set of variables U, a DAG D is called a *Bayesian Network* of P if D is a minimum I-map of P.

There is an elegant method of inferring the minimal I-map of P that is based on the notion of a Markov blanket and a boundary DAG, which are defined next.

Definition 6: A Markov blanket of element $X_i \in U$ is a subset S of U for which $I(X_i, S, U - S - X_i)$ and

$X_i \notin S$. A set is called a Markov *boundary*, B_i of X_i if it is a minimal Markov blanket of X_i, i.e., none of its proper subsets satisfy the triplet independence relation. For example considering the domain U in Figure 1, the random variable X_7, $S = \{X_5, X_6, X_4\}$ is a Markov Blanket which means given the state of $\{X_5, X_6, X_4\}$, X_7 is independent of any other random variable in the domain which is not in S namely X_1 or say X_4. A subset of S, $S_1 = \{X_5, X_6\}$ is a minimal Markov Blanket of X_7 and hence, a Markov Boundary of X_7 which implies that given X_5 and X_6, X_7 is independent of the rest of the random variables in U and any proper subset of S_1 namely singleton sets $\{X_5\}$ or $\{X_6\}$ will not be Markov Blanket for X_7.

Definition 7: Let M be a dependency model defined on a set $U = \{X_1, \cdots, X_n\}$ of elements, and let d be an ordering $\{X_{d1}, X_{d2}, \cdots\}$ of the elements of U. The *boundary strata* of M termed as B_M relative to d is an ordered set of subsets of U, $\{B_{d1}, B_{d2}, \cdots\}$ such that each B_{di} is a Markov boundary (defined above) of X_{di} with respect to the set $U_{di}(\subset U) = \{X_{d1}, X_{d2}, \cdots, X_{d(i-1)}\}$, i.e. B_{di} is the minimal set satisfying $B_{di} \subset U$ and $I(X_{di}, B_{di}, U_{di} - B_{di})$. *The DAG created by designating each B_{di} as the parents of the corresponding vertex X_{di} is called a* boundary DAG *of M relative to d.* It should be noted here that the only ordering restriction is that the variables in the Markov Boundary set (of a particular variable) has to be ordered before the random variable. For the ordering of variables $X_1 \cdots X_9$ in Figure 1, the boundary strata of underlying dependency model M over the domain U is given as

$$B_M = \{\{\phi\}, \{\phi\}, \{\phi\}, \{\phi\}, \{X_1, X_2\}, \{X_3, X_4\}, \{X_5, X_6\}, \{X_4\}, \{X_7, X_8\}\}$$

It is easily observed from the definition that the DAG in Figure 2 is a Boundary DAG corresponding to dependency model M over domain U in Figure 1.

This leads us to the final theorem that relates the Bayesian network to I-maps, which has been proven in [3]. This theorem is the key to constructing a Bayesian network.

Theorem 2: Let M be any dependency model satisfying the axioms of independence listed in Eqs. 3- 5. If the graph structure D is a boundary DAG of M relative to ordering d, then D is a minimal I-map of M. **Proof:** For proof, see [2].

This theorem along with definitions 2, 3, and 4 above, specifies the structure of the Bayesian network. We use these to prove our theorem regarding the structure of Bayesian network to capture the switching activity of a combinational circuit.

Let a combinational circuit consist of gates $\{G_1, \cdots, G_N\}$ with n primary input signals denoted by the set $\{I_1, \cdots, I_n\}$. Let the output of gate G_i be denoted by O_i. The inputs to a gate are either a primary input signal or output of another gate. The switching of these input signal and output lines, $\{I_1, \cdots, I_n, O_1, \cdots, O_N\}$, are the random variables of interest. Note that the set of output lines include both intermediate lines and the final output lines. Let X_i be the switching at the i-th line, which is either an input or an output line, taking on four possible values, $\{x_{00}, x_{01}, x_{10}, x_{11}\}$, corresponding to the possible transitions: $0 \to 0, 0 \to 1, 1 \to 0, 1 \to 1$.

Definition 8: A Logic Induced Directed Acyclic Graph (LIDAG) structure, LD, corresponding to a combinational circuit consists of nodes, X_is, representing the switching at each line and links between them is constructed as follows: The parents of a random variable representing the switching at an output line, O_i, of a gate G_i are the nodes representing switchings at the input lines of that gate. Each input line is either one of $\{I_1, \cdots, I_n\}$ or an output of another gate. The DAG shown in Figure 2 is a LIDAG corresponding to the combinational circuit shown in Figure 1.

Theorem 3: The LIDAG structure, LD, corresponding to the combinational circuit is a minimal I-map of the underlying switching dependency model and hence is a Bayesian network.

Proof: Let us order the random variables, $\{X_1, X_2, \cdots, X_{N+n}\}$ such that (i) variables representing the switching at the input lines appear first followed by those representing the output lines of the gates, and (ii) if a line O_i is an input to a gate whose output line is O_k then the variable corresponding to line O_i appear before O_k.

With respect to this ordering, the Markov boundary of a node, X_i, is given as follows. If X_i represents switching of an input signal line, then its Markov boundary is the null set. And, since the switching of an output line is only dependent on the inputs of the corresponding gate, the Markov boundary of a variable representing an output line consists of just those that represent the inputs to that gate. In the LIDAG structure the parents of each node are its Markov boundary elements hence the LIDAG is a boundary DAG. And, by Theorem 2 listed above the LIDAG is a minimal I-map and thus a Bayesian network (BN).

It is interesting to note that the LIDAG structure corresponds exactly to the DAG structure one would arrive by considering the *principle of causality*, which states that one can arrive at the appropriate Bayesian network structure by directing links from nodes that represent causes to nodes that represent immediate effects [3]. Thus, directed links in the graph denote immediate cause and effect relationship. In a combinational circuit the immediate causes of switching at a line are the switchings at the input lines of the corresponding gate.

IV. Formation of the LIDAG-BN

We first illustrate with an example how switching in a combinational circuit at circuit level can be represented by a LIDAG structured Bayesian network (LIDAG-BN). Then we show how the conditional probabilities that quantify the links of LIDAG-BN are specified.

Let us consider the circuit with fives gates shown in Figure 1. We are interested in the switching at each of the 9 numbered lines in the circuit. Each line can take four values corresponding to the four possible transitions: $\{x_{00}, x_{01}, x_{10}, x_{11}\}$. Note that this way of formulating the random variable effectively models temporal correlation since only first order temporal correlation is exhibited in combinational circuit under zero-delay scenario [15]. To capture all higher order spatial correlations, we form the interconnection (through edges) and quantify them by the conditional probabilities for the child-parent group in the LIDAG. The probability of switching at a line would be given by $P(X_i = x_{01}) + P(X_i = x_{10})$[2]. The LIDAG structure for the circuit is shown in Figure 2. Dependence among the nodes that are not connected directly is implicit in the network structures. For example, nodes X_1 and X_2 are independent of each other, however, they are *conditionally* dependent given the value of say node X_9. Or the transition at line 5, X_5, is dependent on the transitions at lines 1 and 2, represented by the random variables, X_1 and X_2, respectively. Thus, the transitions of line 5 are *conditionally independent* of all transitions at other lines *given* the transition states of lines 1 and 2.

The joint probability function that is modeled by a Bayesian network can be expressed as the product of the conditional probabilities. We also say that joint probability function p factorizes in the Bayesian Network. For the Bayesian network structure in Fig. 2 the corresponding joint probability density is given by the following factored form. It has to be noted that this factored form can only be obtained for circuits without a feedback.

$$P(x_1, \cdots, x_N) = \prod_v P(x_v | x_{\text{parent}(v)}) \qquad (6)$$

$$\begin{aligned} P(x_1, \cdots x_9) \quad &= P(x_9|x_7, x_8)P(x_8|x_4)P(x_7|x_5, x_6) \\ &\quad P(x_6|x_3, x_4)P(x_5|x_1, x_2)P(x_4) \\ &\quad P(x_3)P(x_2)P(x_1) \end{aligned}$$

The conditional probabilities of the lines that are directly connected by a gate can be obtained knowing the type of the gate. For example, $P(X_5 = x_{01}|X_1 = x_{01}, X_2 = x_{00})$ will be always 1 because if one of the inputs of an OR gate makes a transition from 0 to 1 and the other stays at 0 then the output always makes a transition from 0 to 1. A complete specification of the conditional probability of $P(x_5|x_1, x_2)$ will have 4^3 entries since each variable has 4 states. These conditional probability specifications are determined by the gate type. Thus, for an AND gate, if one input switches from 0 to 1 and the other from 1 to 0, the output remains at 0. By specifying a detailed conditional probability we ensure that the spatio-temporal effect (first order temporal and higher order spatial) of any node are effectively modeled.

[2]Probability of the event $X_i = x_i$ will be denoted simply by $P(x_i)$ or by $P(X_i = x_i)$.

The last four terms in the right hand side of Eq. 7 represent the statistics of the input lines. Given the statistics of the input lines, we would like to infer the probabilities of all the other nodes. A brute force way of achieving this would be to compute the marginal probabilities by summing over possible states, thus, $P(x_9, x_1) = \sum_{x_2, \cdots, x_8} P(x_1, \cdots, x_9)$. This, obviously, is computationally very expensive and, in addition, does not scale well. In the next section, we show how the structure of the Bayesian network can be used to efficiently compute the required probabilities.

V. Computations using a Bayesian Network

It would be computationally convenient if we could compute probabilities in a Bayesian network by local message passing. Unfortunately, we cannot directly update a Bayesian network by local message passing if the underlying undirected graph structure has cycles. So, the first step of the computation strategy is to transform the original DAG structure into a undirected tree structure whose nodes are subsets of the original random variables [2]. Such a tree is referred to as the *junction tree of cliques of random variables* and the process is referred to as the Bayesian network compilation process. After the compilation process, the required probabilities can be easily computed by local message passing in a simple propagation algorithm. In this section, we will first outline the steps involved in the compilation process and then sketch the propagation algorithm. We should point out to the reader that there are many Bayesian network software that will automatically do these steps. For this study we used the HUGIN Bayesian network software for the implementation of inference engine which performs the steps discussed throughout the remaining section [21].

A. Network Compilation

The compilation process ensures two aspects: 1) it builds a tree of cliques which implies that there is no loop in the tree, and 2) The tree is built in a special fashion to help a local message passing scheme described in Section V-B. The first step of the compilation process is the formation of the corresponding

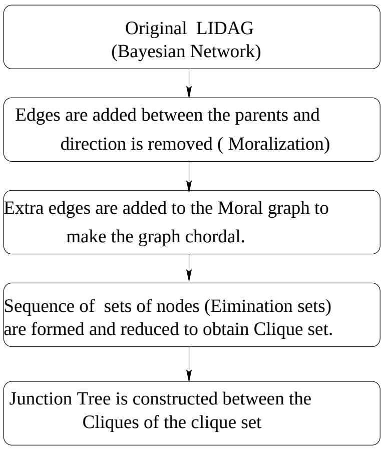

Fig. 3. Network Compilation steps

moral graph. This is followed by triangularization of the Moral graph. The clique set is then identified and a junction tree of cliques is formed. These steps are shown as a flow-chart in Figure 3.

A.1 The Moral Graph

The first step of the compilation process is to create an undirected graph structure called the *moral graph* (denoted by D^m) given the Bayesian network DAG structure (denoted here by D). The moral graph represents the Markov structure of the underlying joint function [2]. If probability function P factorizes recursively like that shown in Equation 7 in a DAG D, it also can be expressed in a factorized form in the moral graph of D [2]. From a DAG, which is the structure of a Bayesian network, a moral graph is obtained by adding undirected edges between the parents of a common child node and dropping the directions of the links (shown in dashed line in Figure 4). This step ensures that every parent child set is a complete subgraph. Moral graph is undirected and due to the added links, some of the independencies displayed in DAG will not be graphically visible in moral graph. Some of the independencies that are lost in the transformation contributes to the increased computational efficiency but does not affect the accuracy [2]. The independencies that are graphically seen in the moral graph are used in inference process to ensure local message passing.

A.2 The Chordal Graph

This step is the triangulation of the moral graph to make it chordal (often referred to as triangulated). We add additional links to the moral graph so that all cycles longer than 3 nodes are broken into cycles of three nodes. In our example, the dash-dotted line between X_4 and X_7 in Figure 4 is added to the moral graph to triangulate it. One can always add all possible links between nodes to make the graph chordal,

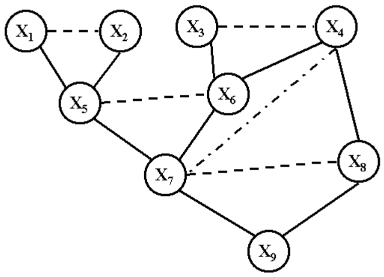

Fig. 4. Triangulated undirected graph structure that encodes that same dependencies as the DAG structure shown in Figure 2. The solid links are from the Bayesian network. The dashed links are added during moral graph construction. And, the dot-dashed link is added during the final triangulation step.

transforming the moral graph into a complete graph, however, the challenge is to make a graph chordal using a minimum number of additional links. This problem is in general NP-hard. Hence heuristics are used. A sub-optimal solution to find the minimum number of links may hamper computational efficiency but the resultant graph will be a chordal one, thus maintaining the accuracy of the inference process. In HUGIN, a minimum fill-in edges heuristics is used [2].

A.3 The Junction Tree

Definition 9: A **junction tree** T of cliques is a tree with nodes representing cliques (collection of completely connected nodes) and between any two cliques C_i and C_j in the tree T there is a unique path. Also, the elements in the intersection set $C_i \cap C_j$ are present in all the cliques in that unique path between C_i and C_j. This property in junction tree is also referred to as the *running intersection property.*

The existence of at least one junction tree in a chordal graph is guaranteed [2]. The running intersection property of a junction tree is an important one and is utilized in the local message passing based updating algorithm described later. It is worth noting that it is possible to construct a tree of node cliques from the moral graph itself, but it will not have the running intersection property and hence it will not be suitable for local message passing based updating.

A.4 From Moral Graph to Junction Tree

In this section, we outline the details how a moral graph is transformed into a junction tree. The process essentially consists of three following steps.

• **Step 1: Moral Graph to Elimination Set**
All the vertices of the moral graph are first unnumbered. An unnumbered vertex that needs the minimum number of additional edges between its neighbors (considering both numbered or unnumbered neighbors) is chosen. This vertex is then labeled with the highest available node number, say i, starting from a number equal to the total number of nodes. A set

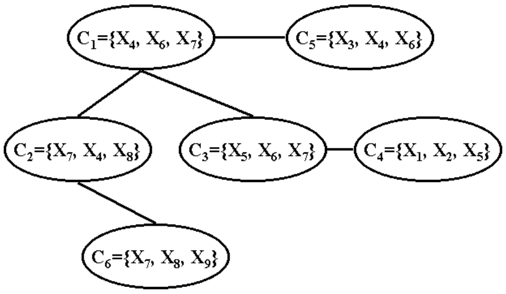

Fig. 5. Junction tree of cliques.

C_i is then formed consisting of the selected vertex and its still unnumbered neighbors. Edges are filled in between any two unlinked nodes in this set C_i. Then the maximum available node number i is decremented by 1. This process is repeated until there is no unnumbered nodes. The resultant graph is now a chordal one. The generated sequence of cliques C_i's is termed the *elimination set* of cliques of the graph, subset of which is used to construct the junction tree.

Due to the way the cliques in the elimination set are derived, they will possess the running intersection property [2]. It can also be shown that the set of cliques in the final junction tree of a graph is subset of the elimination set [2].

In our example moral graph, node X_9 is first selected since no fill-in edge is needed because all the neighbors (remember the moral graph is undirected) are already linked. This node X_9 is assigned the number 9 - the total number of nodes in the graph. The set C_9 is then formed by nodes $\{X_9, X_8, X_7\}$. The nodes X_8 and X_7 are not yet numbered. For the second cycle, the nodes X_8, X_7, X_6, and X_4 cannot be selected as they each would require one fill-in edges amongst its neighbors, whereas the neighbors of X_3 does not require any fill-in edges. Hence X_3 is numbered 8 in our example and C_8 is formed by $\{X_3, X_4, X_6\}$. For the third cycle, we then select X_2, numbering it as 7 and forming $C_7 = \{X_2, X_1, X_5\}$. In the fourth cycle, node X_1 is assigned 6 and $C_6 = \{X_1\}$ is formed. We then select X_5, assign a number 5, and form $C_5 = \{X_5, X_6, X_7\}$. Node X_8 is assigned number 4, and $C_4 = \{X_8, X_7, X_4\}$ is formed. In this step, a fill-in edge between X_4 and X_7 is added. We then assign the number 3 to X_7, the number 2 to X_6, and the number 1 to X_4. Thus, we form $C_3 = \{X_7, X_6, X_4\}$, $C_2 = \{X_6\}$ and $C_1 = \{X_4\}$.

• **Step 2: Reducing the Elimination Set**

It is shown in [2] that if $C_1, \cdots C_k$ is a sequence of sets having running intersection property and $C_t \subseteq C_p$ for some $t \neq p$ then the ordered set $C^{'} = \{C_1, \cdots C_{t-1}, C_p, C_{t+1}, \cdots, C_{p-1}, C_{p+1}, C_k\}$ also has running intersection property. By this property C_t can be eliminated for all $C_t \subseteq C_p, p \neq t$. Hence, the elimination set can be reduced to obtain the minimal ordered set of cliques called *Clique set* representing the chordal graph completely. This ordering of cliques is guaranteed to possess the running intersection property.

• **Step 3: Elimination Set to Junction Tree**

For each clique C_i of the Clique set ordered to have running intersection property, we have to add a link to clique C_j where j is **any one** of the set $\{1, 2, \cdots, i-1\}$ such that $C_i \cap (C_1 \cup C_2 \cup \cdots \cup C_{i-1}) \subseteq C_j$

Figure 5 shows the junction tree for our running example. It can be observed that the cliques in our example are $C_1 = \{X_4, X_6, X_7\}$, $C_2 = \{X_7, X_4, X_8\}$, $C_3 = \{X_5, X_6, X_7\}$, $C_4 = \{X_1, X_2, X_5\}$, $C_5 = \{X_3, X_4, X_6\}$, and $C_6 = \{X_7, X_8, X_9\}$. This cliques are obtained by reducing the elimination set. The edges between them are formed the algorithm in step 3. For our example in Figure 5, for the clique C_6, we have to find the set that contains the following intersection set

$$\begin{aligned} C_6 \cap (C_1 \cup C_2 \cup \cdots \cup C_6) = \\ \{\{X_7, X_8, X_9\} \cap \{X_4, X_6, X_7, X_8, X_5, X_1, X_2, X_3\}\} \\ = \{X_7, X_8\} \end{aligned}$$

It is obvious that $\{X_7, X_8\} \subseteq C_2$, hence we add a link between C_6 and C_2. Every clique with an edge between them will have a non-null set of nodes common between them, which is referred to as the *separator set*. These common variables play key role in evidence propagation. If there were two cliques that are not connected by an edge and still have common variables then these variables must be present in all the cliques in between the unique path between the two cliques to guarantee correct probability values for the random variables of the entire network while probabilities are updated by local propagation. As it can be seen that C_3 and C_6 both contain X_7 and X_7 is also present in all the cliques in the path from C_3 to C_6 namely in C_1 and C_2. This helps to preserve probabilities of the random variables of the entire network while updating an evidence by local message passing.

B. *Propagation in Junction Trees*

Once the junction tree of cliques is formed in the compilation process, we need to form the distribution function for the cliques. We then follow a two phase algorithm for message propagation. All the messages propagated are between neighboring cliques. These steps are shown in Figure 6.

It can be proven that the dependency properties of a DAG, which carry over to moral graphs, also are preserved in the triangulated moral graph [2]. The joint probability function that factorizes on the moral graph will also do so on the triangulated one since each clique in the moral graph is either a clique in

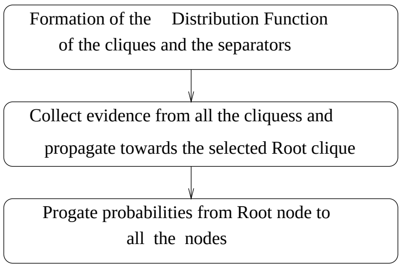

Fig. 6. Steps involved in propagation of evidence in the junction tree

this graph or subset of a clique. Let $\{\mathbf{x_c}\}$ be the set of nodes in clique c in the junction tree. The joint probability function over these variables is denoted by $p(\mathbf{x_c})$. Let $\{\mathbf{x_s}\}$ be the set of nodes in a separator set s between two cliques in the junction tree. The joint probability function over these variables is denoted by $p(\mathbf{x_s})$. Let CS denote the set of all cliques and SS denote the set of separators in the junction tree. The joint probability function factorizes over the junction tree in the following form [2]:

$$p(x_1, \cdots, x_N) = \prod_{c \in \mathrm{CS}} p(\mathbf{x_c}) / \prod_{s \in \mathrm{SS}} p(\mathbf{x_s}) \qquad (7)$$

A separator set s is contained in two neighboring cliques, c_1 and c_2. If we associate each of the product terms over the separators in the denominator with one its two neighboring cliques, say c_1, then we can write the joint probability function in a pure product form as follows. Let $\phi_{c_1}(\mathbf{x_{c_1}}) = p(\mathbf{x_{c_1}})/p(\mathbf{x_s})$ and $\phi_{c_2}(\mathbf{x_{c_2}}) = p(\mathbf{x_{c_2}})$, then the joint probability function as expressed as:

$$P(x_1, \cdots, x_N) = \prod_{c \in \mathrm{CS}} \phi_c(\mathbf{x_c}) \qquad (8)$$

where the factors $\phi_c(\mathbf{x_c})$ are also commonly referred to as the potential function over the nodes $\{x_c\}$ in clique c, and CS is the set of cliques. These functions, $\phi_c(\mathbf{x_c})$s, can be formed by multiplying the conditional probabilities, from the input Bayesian network specification, of nodes in the clique c.

Let us now focus on the information flow in the neighboring cliques to understand the key feature of the Bayesian updating scheme. As shown in Figure 7, let two cliques A and B have probability potentials ϕ_A and ϕ_B, respectively. Let S be the set of nodes that separates cliques A and B. Let us say that there is new evidence for some node. This will change the probabilities of all the other nodes such that two neighboring cliques agree on probabilities on the node set S which is their separator. To achieve this we first compute the marginal probability of S from probability potential of clique A and then use that to scale the probability potential of B as captured by Eq. 10. To achieve this we need to transmit

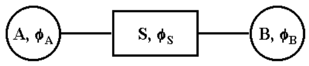

Fig. 7. Two cliques of junction tree along with the separator set.

the scaling factor along the link and this process is referred to as message passing. We have to repeat this process in the reverse direction by computing the marginal probability of S from probability potential of clique B and then use that to scale the probability potential of A. This will ensure that evidence at both the cliques are taken into account. New evidence is absorbed into the network by passing such local messages. The pattern of the message is such that the process is multi-threadable and partially parallelizable. Because the junction tree has no cycles, messages along each branch can be treated independently of the others.

$$\phi_S^* = \sum_{X \in A, X \notin S} \phi_A \qquad (9)$$

$$\phi_B^* = \phi_B \frac{\phi_S^*}{\phi_S} \qquad (10)$$

It would appear that we will need to initiate a message passing over the whole network for each new evidence, however, this need not be the case. In fact, there is a two phase message passing scheme that can integrate all new evidence in two passes. In this scheme, a clique is selected from junction tree to be the root node. In the first phase, the collection phase, all the leaf nodes of the tree send messages towards the root node, which are re-computed according at Eqs. 9 and 10 at each node along the way. Once, the messages from all the leaf nodes have been received, the second phase is initiated when messages are passed back from the root clique towards the leaves. Note that this ensures that along any one link, we have messages along both directions, thus, ensuring all nodes have been updated based on information from all the new evidence.

Junction tree based probabilistic propagation thus serve two purposes: 1) they solve the problem of updating probabilities in a loop and 2) the cliques having running intersection property so updating is performed locally from one clique to the neighboring clique just once. In the next section we discuss the experiments and results obtained for benchmark circuits.

VI. Experimental Results

We mapped 14 ISCAS and 5 MCNC circuits to their corresponding LIDAG structured Bayesian Networks. The conditional probabilities are pre-determined by the type of gate connecting the parents and the child. We have already discussed in Section IV that each node in Bayesian network represents switching at a line in the circuit and can be in one of the four states

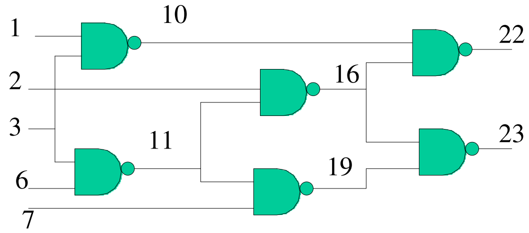

Fig. 8. Combinational circuit c17

$(x_{00}, x_{01}, x_{10}, x_{11})$. Conditional probabilities are assigned by the logical relationship of the gate. In combinational circuits, it is fairly common to encounter gates with a large number of inputs. This may be a problem in terms of the conditional probability table size which grows exponentially with the number of inputs. We solve this problem by adding a few two-input dummy nodes still preserving the logic structure. The switching activity on the original nodes are modeled exactly by this method. We do not consider the switching activity on dummy nodes for the error estimates. For example a four input NAND gate is modeled as two AND gates followed by a NAND gate. We ignore switching activity on the outputs of the first two AND gate and the inputs of the last NAND gate. Thus the switching estimates of the terminals of the original NAND is captured accurately without adding switching activity on dummy nodes. We used HUGIN's Bayesian Network tool for compiling the junction tree and propagating the probabilities. We also performed logic simulation providing "ground truth" estimates of switching.

A. A small example

Let us tabulate the switching activity estimation of each node of c17 ISCAS benchmark. As it can be seen 8 that the nodes 10 and 11 are highly correlated and the nodes 16 and 19 are highly correlated to each other. Since the nodes 10 and 16 both are dependent on node 3 they are correlated too. Table I shows all the circuit nodes under random input sequences and under biased sequences. As it can be observed that in both cases, we have extremely accurate estimates on all nodes of the circuit.

B. Random Inputs

We tabulate the results of switching activity estimation at circuit level by the LIDAG structured Bayesian networks in Table II and in Table III. We also performed simulation with pseudo-random inputs for comparison of the switching estimates. In Table II, we present the circuits where we model the circuit as a single BN without segmentation.

It is evident that if the circuit is considerably large or if the circuit has high connectivity between nodes, then, the size of the cliques formed becomes large. This results in large probability potential tables and might make handling large circuits in one stage impossible with limited memory resources. To handle large

TABLE I

COMPARISON OF ESTIMATED SWITCHING ACTIVITY BY BN MODELING AND SIMULATED SWITCHING ACTIVITY OF EACH NODE OF BENCHMARK C17

nodes of	Random Inputs		Biased Inputs	
c17	Est. Sw_{BN}	Sim. Sw	Est. Sw_{BN}	Sim. Sw
1	0.5	0.499	0.4	0.401
2	0.5	0.500	0.4	0.402
3	0.5	0.499	0.4	0.4
6	0.5	0.5	0.4	0.4
7	0.5	0.5	0.4	0.4
8	0.5	0.499	0.4	0.4
9	0.5	0.499	0.4	0.4
10	0.375	0.375	0.160	0.160
11	0.375	0.374	0.160	0.160
14	0.375	0.374	0.160	0.160
15	0.375	0.374	0.160	0.160
16	0.469	0.469	0.380	0.382
19	0.469	0.469	0.380	0.379
20	0.469	0.469	0.380	0.382
21	0.469	0.469	0.380	0.382
22	0.492	0.493	0.435	0.436
23	0.492	0.492	0.480	0.481

circuits, we adopt a divide and conquer strategy. We first segment the large circuit into smaller ones and then estimate switching activity in them by ensuring consistency of singleton probability of the nodes at the boundary. In Table III, we present the circuits where we model the circuit as multiple BNs. In both the tables, columns 2 and 3 provide the mean and standard deviation of the error of switching activity, which is the error between the switching activity estimated and the switching activity obtained through simulation on each node, respectively. Moreover, we present the % error between the average switching activity estimated over all the nodes with average switching activity through simulation in column 4 (This column is presented separately from column 2 to indicate the % error). The elapsed total time for Bayesian Network based estimation, which is time to compile and time to propagate, is shown in column 5 . We separately show the time for propagation of evidence in column 6. As it can be observed, the propagation time is extremely low irrespective of the size of the circuits.

The platform used here is a single processor DELL PC with 250 Meg RAM and 450 MHz clock speed. We want to emphasize that the time estimates are obtained from a WINDOW based timing feature (function ftime in VISUAL C++) which provides the total time spent in a function module which includes memory access, I/O time along with CPU time. In fact, while handling larger circuit, CPU time is an inadequate indicator in terms of the actual time due to high memory accesses by the estimation algorithms.

As it can be observed from Table II, the circuits which are small (namely c17, comp, count, pcler8 etc.)

TABLE II

Experimental results on switching activity estimation by single Bayesian network models for ISCAS'85 and MCNC'89 benchmark circuits for random input sequences.

Circuits	Error statistics over each node		Overall % Error and time for all nodes		
	μ_{Err}	σ_{Err}	% Error	Elapsed Time (s) Total	Update
c17	0.0002	0.0004	0.02%	<.001	<.001
max_flat	0.0004	0.0005	0.10%	<0.001	<.001
voter	0.0002	0.0005	0.04%	0.11	<.001
count	0.0001	0.0006	0.03%	0.33	<.001
comp	0.0000	0.0003	0.00%	0.22	<.001
pcler8	0.0002	0.0007	0.03%	0.11	<.001

TABLE III

Experimental results on switching activity estimation by single Bayesian network models for ISCAS'85 and MCNC'89 benchmark circuits for random input sequences.

Circuits	Error statistics over each node		Overall % Error and time for all nodes		
	μ_{Err}	σ_{Err}	% Error	Elapsed Time (s) Total	Update
b9	0.0004	0.0020	0.10%	0.33	<.001
c8	0.0002	0.0017	0.05%	0.93	<.001
alu4	0.0001	0.0198	0.86%	0.54	<.001
malu4	0.0011	0.0204	0.23%	0.21	<.001
c432	0.0111	0.0300	2.08%	1.33	<.001
c499	0.0002	0.0038	0.09%	0.33	<.001
c880	0.0012	0.0088	0.57%	1.24	<.001
c1355	0.0015	0.0189	0.41%	2.29	<.001
c1908	0.0009	0.0090	0.22%	6.09	<.001
c3540	0.0029	0.0400	0.79%	8.62	<.001
c5315	0.0035	0.0269	0.80%	15.08	<.001
c6288	0.0140	0.0465	3.34%	18.70	<.001
c7552	0.0031	0.0460	0.70%	18.26	<.001

and so do not require any segmentation, the Bayesian Network is extremely accurate and the standard deviation of error calculated on each node is extremely low indicating that switching at every node is estimated accurately. The estimation time is low. The circuits which were segmented shown in Table III have standard deviation of the order of 0.02. The maximum error that is encountered is around 3.5% for c6288. The minimum error is zero. **Out of the 19 benchmark circuits 17 of them had less than 1% error and 2 of them have an error between 1% and 3.5%**. It can also be observed that updating and propagating all the input evidences in almost all the circuits were performed in the order of 1 milliseconds. This is advantageous particularly if the switching activity has to be estimated for different input signal statistics. The circuits can be pre-compiled, only propagation has to be done for different input statistics. We found that BN is an accurate model of switching activity. We analyzed the circuits that showed standard deviation above 0.03 (namely c6288 and c7552) and we observed that errors occurred for some nodes in the segment boundaries. We are currently investigating an accurate way to handle segmentation so that errors do not propagate through the segments. For all the large circuits including c5315, c6288, c7552, we have very low mean error which is encouraging considering the size of the circuit. With accurate segmentation, we can achieve very high accuracy uniformly throughout the circuit with considerably *low elapsed time*.

VII. Conclusions and Future Directions

This paper introduces a switching activity estimation tool that encapsulates all the dependencies, both in the internal nodes and in the inputs, in reasonable time and with high accuracies. We have shown results of the estimated switching activity for pseudo-random inputs and biased inputs. The results are very competitive in terms of accuracy and the elapsed time for estimation. The model handles spatio-temporal dependencies between the nodes. It also models conditional dependencies of the nodes. The proposed model is accurate and captures higher order correlation among lines and is not restricted to pair-wise correlations. Our future effort focuses on the segmentation techniques, and input modeling for capturing spatial correlation at the primary inputs using the same BN model.

Acknowledgments

This work is conducted in the Department of Computer Science and Engineering, University of South Florida and is supported in part by a National Science Foundation Grant No. CCR-0098103.

References

[1] S. Bhanja, N. Ranganathan, "Dependency Preserving Probabilistic Modeling of Switching Activity using Bayesian Networks", to appear in *38th Design Automation Conference*, 2001.

[2] R. G. Cowell, A. P. David, S. L. Lauritzen, D. J. Spiegelhalter, "Probabilistic Networks and Expert Systems", Springer-Verlag New York, Inc., 1999.

[3] J. Pearl, "Probabilistic Reasoning in Intelligent Systems: Network of Plausible Inference", Morgan Kaufmann Publishers, Inc., 1988.

[4] A. Ghosh, S. Devadas, K. Keutzer, and J. White, "Estimation of Average Switching Activity in Combinational and Sequential Circuits", *Proceedings of the 29th Design Automation Conference*, pp.253-259, June 1992.

[5] R. Marculescu, D. Marculescu, and M. Pedram, "Probabilistic Modeling of Dependencies During Switching Activity Analysis", *revised version submitted to IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems*, URL `http://atrak.usc.edu/~massoud/sign_download.cgi?pecp-journal.ps`

[6] R. Marculescu, D. Marculescu, and M. Pedram, "Probabilistic Modeling of Dependencies During Switching Activity Analysis", *IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems*, vol 17-2, pp.73-83, February 1998.

[7] R. E. Bryant, "Symbolic Boolean Manipulation with Ordered Binary-Decision Diagrams", *ACM Computing Surveys*, Vol. 24, No. 3, pp. 293-318, Sept. 1992.

TABLE IV

Comparison of CPU time and percentage error of different estimation techniques. Elapsed time given in the last column includes CPU, memory and I/O time.

	Marculescu et. al., 94[19]			Schneider et. al., 96*[16]			Marculescu et. al., 98*[5]			Bayesian Networks		
Circuit	μ_{Err}	σ_{Err}	Time CPU(s)	μ_{Err}	σ_{Err}	Time CPU (s)	μ_{Err}	σ_{Err}	Time CPU(s)	μ_{Err}	σ_{Err}	Time Elapsed(s)
c17	0.012	0.02	0.35	-	-	-	0.012	0.02	0.04	0	0	<0.001
c432	0.013	0.02	276.43	0.016	-	11	0.028	0.04	11.82	0.011	0.03	1.33
c499	0.005	0.01	519.56	-	-	-	0.013	0.01	10.57	0	0	0.33
c880	0.016	0.03	320.11	0.006	-	34	0.013	0.02	18.35	0.001	0.01	1.24
c1355	0.003	0	330.11	0.005	-	11	0.004	0	4.24	0.001	0.02	2.29
: c1908	0.004	0.01	489.23	0.01	-	8	0.009	0.02	12.69	0.001	0.01	6.09
c3540	0.028	0.03	3307.11	0.014	-	65	0.03	0.04	60.86	0.003	0.04	8.62
c5315	-	-	-	-	-	-	-	-	-	0.003	0.03	15.08
c6288	-	-	4530	0.023	-	126	0.014	0.02	29.1	0.014	0.05	18.7
c7552	-	-	-	-	-	-	-	-	-	0.003	0.05	18.26

[8] F. N. Najm, "Transition Density: A New Measure of Activity in Digital Circuits", *IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems*, vol 12-2, pp.310-323, February 1993.

[9] S. Ercolani, M. Favalli, M. Damiani, P. Olivo, and B. Ricco, "Testability Measures in Pseudorandom Testing", *IEEE Transactions on CAD*, vol.11, pp. 794-800, June 1992.

[10] C.-S. Ding, C.-Y. Tsui, and M. Pedram, "Gate-Level Power Estimation Using Tagged Probabilistic Simulation", *IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems*, vol.17-11, pp.1099-1107, November, 1998.

[11] K. Parker, and E. J. McCluskey, "Probabilistic Treatment of General Combinational Networks", *IEEE Trans. on Computers*, vol. C-24, pp.668-670, June 1975.

[12] B. Kapoor, "Improving the Accuracy of Circuit Activity Measurement", *Proc. ACM/IEEE Design Automation Conference*, pp. 734-739, June 1994.

[13] C.-Y. Tsui, M. Pedram and A. M. Despain,"Efficient Estimation of Dynamic Power Dissipation with an Application", *Proc. ACM/IEEE Design Automation Conference*, pp. 224-228, November 1993.

[14] F. N. Najm, R. Burch, P. Yang, and I. N. Hajj, " Probabilistic Simulation for Reliability Analysis of CMOS Circuits", *IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems*, vol 9-4, pp.439-450, April 1990.

[15] P. Schneider, and U. Schlichtmann, "Decomposition of Boolean Functions for Low Power Based on a New Power Estimation Technique", *Proc. 1994 Int'l Workshop on Low Power Design*, pp.123-128, April 1994.

[16] P. H. Schneider, U. Schlichtmann, and B. Wurth, "Fast power estimation of large circuits", *IEEE Design & Test of Computers*, vol. 13-1, pp. 70-78, Spring 1996.

[17] F. N. Najm and M. G. Xakellis, "Statistical estimation of the switching activity in VLSI circuits," *VLSI Design*, vol. 7-3, pp. 243-254, 1998

[18] A. Murugavel, N. Ranganathan, R. Chandramouli and S. Chavali, "Least Square Estimation of Average Power in Digital CMOS Circuits", *IEEE Transactions on VLSI Systems*, Vol. 10, No. 1, pp. 55-58, Feb 2002.

[19] R. Marculescu, D. Marculescu, and M. Pedram, "Switching Activity Analysis Considering Spatiotemporal Correlations", *Proc. 1994 Intl. Conference on Computer Aided Design*,pp.294-299, November 1994.

[20] J. M. Rabaey, M. Pedram, "Low Power Design Methodologies", Kluwer Academic Publishers, 1996.

[21] `URL http://www.hugin.com/`

Sanjukta Bhanja received her BE and MSc(engg.) in EE from Jadavpur University and Indian Institute of Science, Bangalore, respectively. She finished her Ph.D in Computer Science and Engineering from University of South Florida, Tampa. She is currently an assistant Professor in the Department of Electrical Engineering at the University of South Florida, Tampa. Her research interests includes design automation, low power VLSI design, power estimation, optimization, system-on-a-chip, probabilistic modeling, VLSI testing and expert systems based modeling.

N. Ranganathan (S'81-M'88-SM'92, F'02) received the B.E. (Honors) degree in electrical and electronics Engineering from Regional Engineering College, Tiruchirapalli, University of Madras, India in 1983, and the Ph.D. degree in Computer Science from the University of Central Florida, Orlando in 1988. He is currently a professor in the Department of Computer Science and Engineering and the Nanomaterials and Nanoelectronics Research Center at the University of South Florida, Tampa. His research interests include VLSI system design, design automation, power estimation and optimization, high level synthesis, embedded systems and computer architecture. He has developed many special purpose VLSI chips for computer vision, image processing, pattern recognition, data compression and signal processing applications. He has published over 175 papers in reputed journals and conferences and is a co-owner of five U.S. patents. He was elected as Fellow of IEEE for his contributions to VLSI systems design. He has served on the program committees of several conferences and on the editorial boards of *IEEE Transactions on VLSI Systems*, *IEEE Transactions on Circuits and Systems TCAS-II*, *IEEE Transactions on CAS for Video Technology*.