Stimulus-Free RT Level Power Model using Belief Propagation

by

Sathishkumar Ponraj

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Electrical Engineering
Department of Electrical Engineering
College of Engineering
University of South Florida

Major Professor: Sanjukta Bhanja, Ph.D.
Wilfrido A. Moreno, Ph.D.
Kenneth A. Buckle, Ph.D.

Date of Approval:
October 25, 2004

Keywords: Top-down, Bottom-up, Register Transfer Level, Behavioral Level, Simulation, Clique, Inference, Sampling

DEDICATION

To my parents, Ponraj & Hemalatha, for their unconditional love, guidance, support and enthusiasm.

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my sincere gratitude to my advisor Dr. Sanjukta Bhanja for her constant encouragement, support and guidance throughtout the course of the research. I would also like to thank Dr. Wilfido A. Moreno and Dr. Kenneth A. Buckle for accepting my request and serving my committee. I would also like to thank my friend Karthikeyan Balakrishnan for encouraging me at hard times and giving me innovative ideas.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

STIMULUS-FREE RT LEVEL POWER MODEL USING BELIEF PROPAGATION

Sathishkumar Ponraj

ABSTRACT

Power consumption is one of the major bottleneck in current and future VLSI design. Early microprocessors which consumed a few tens of watts are now replaced by millions of transistors and with the introduction of easy-to-design tools to explore at unbelievable minimum dimensions, increase in chip density is increasing at a alarming rate necessitates faster power estimation methods. Gate level power estimation techniques are highly accurate methods but when time is the main constraint power has to be estimated a lot higher in the abstraction level. Estimating power at higher levels also saves valuable time and cost involved in redesigning when design specifications not met. We estimate power at every levels of abstraction for a breadth first design-space exploration. This work targets a stimulus-free pattern-insensitive RT level hierarchical probabilistic model, called Behavioral Induced Directed Acyclic Graph (BIDAG), that can freely traverse between the RT and logic level and we prove that such a model corresponds to a Bayesian Network to map all the dependencies and can be used to model the joint probability distribution of a set of variables. Each node or variable in this structure represents a gate level Directed Acyclic Graph structure, called the Logic Induced Directed Acyclic Graph (LIDAG). We employ bayesian networks for the exact representation of underlying probabilistic framework at RT level capturing the dependence exactly and again use the same probabilistic model for the logic level. Bayesian networks are graphical representations used to concisely represent the uncertain knowledge of the system. In order to get an posterior belief of a query node or variable, with or without preset nodes or variables called the evidence nodes, we use stochastic inference algorithm, based on importance sampling method, called the Evidence Pre-propagation Importance Sampling(EPIS) which is anytime and scales really well for RT and logic networks. Experimenatal results indicate that this method of estimation yields

high accuracy and is qualitatively superior to macro-models under a wider range of input patterns. The main highlights of this work is that as it is probabilistic model, it is *input pattern independent* and nonsimulative property implies less time for power modelling.

CHAPTER 1

INTRODUCTION

Until recently the major concerns of the VLSI design research were area, performance, cost and reliability; the design power was of secondary concern. With the device sizes gradually shrinking to have better performance and packaging of millions of transistors in a single chip the power dissipation, which increases as a result, has become a more critical design concern than speed and area. The increase in power dissipation increases the cost for cooling the chip and thereby increases the battery weight. To overcome these problems designers had to make devices that operate at low power. Estimation of power consumption of the design is the first step towards integrating power minimization techniques.

1.1 VLSI Power Consumption

The average power dissipated in a digital CMOS circuit is given by the formula,

$$P_{avg} = P_{dynamic} + P_{short-circuit} + P_{leakage} + P_{static} \tag{1.1}$$

where, P_{avg} is the average power dissipation, $P_{dynamic}$ is the dynamic power dissipation due to switching of transistors, $P_{short-circuit}$ is the short-circuit current power dissipation when there is a direct path from power supply to ground, $P_{leakage}$ is the power dissipation due to leakage currents and P_{static} is the static power dissipation.

1.1.1 Static Power Consumption

The CMOS circuits are designed in such a way that at any time for a input state only one of the two networks, either the pull-up or the pull-down network is ”ON” and the other network is ”OFF”.

But in reality, there is some small static dissipation due to the reverse bias leakage between diffusion regions and the substrate. This subthreshold conduction contributes to the static dissipation.

1.1.2 Short-Circuit Power Consumption

The current that flows through both the pull-up and pull-down networks during the transition period, i_{cc}, is called the short-circuit current and hence the short circuit power dissipation, $P_{short-circuit}$.

1.1.3 Dynamic Power Consumption

The charging and discharging of the parasitic capacitance contributes to the dynamic power dissipation, $P_{dynamic}$. The dynamic power has been the dominant component in power dissipation and contributes to about 80% of the total power. The dynamic power is given by the equation

$$P_{dynamic} = 0.5 f_{clk} V_{dd}^2 C_l S(x) \tag{1.2}$$

where $P_{dynamic}$ is the dynamic power dissipation, f_{clk} is the clock frequency, V_{dd} is the supply voltage, C_l is the load capacitance and $S(x)$ is the average switching activity of an output node x. It is evident from equation 1.2 that except for the load capacitance and the switching activity of a node, all the other variables are constants. Hence the dynamic power estimation can be easily calculated if the load capacitance and switching activity are known.

Power estimation has been performed at differnt levels of of abstraction, namely architectural level, algorithmic level, functional block level, logic level, circuit level. Reader can refer to the Chapter 2 for clear understanding of various power estimation techniques at each of these abstraction levels. Power can be accurately estimated at low level approaches, such as the logic level and the circuit level but the only problem with the power estimation at the latter levels is that by the time the circuit has been specified in gates and transistors, if the design specifications are not met, the whole circuit has to be designed again with different implementation. Hence the time involved and the heavy cost for redesign step act as driving forces towards the high level power estimation. Though power is a little inaccurate in this level, the designer is allowed to perform architectural exploration and try different implementations lot early in the design cycle. In this thesis we bring in

to focus the importance of RT level power estimation and propose a technique of power estimation at RT level using a Probabilistic Graphical model called the*Bayesian networks.* Bayesian networks are discussed in detail in Chapter 3. At the RT level if the gate level implementation of the circuit is known the dynamic power, which is the dominating power component is completely dependent on the switching activity of the node. Switchin activity of a node denotes to the actual activity of the node, when the node makes a transition from $0 \rightarrow 1$ or $1 \rightarrow 0$. Higher switching activity means more number of node transitions and more dynamic power dissipation. To effectively model the switching activity of a node the following factors should be taken in to concern are node correlation, input statistics, circuit connectivity. The node correlation is split in to two sub components, namely spatial correlations and temporal correlations. The *Temporal correlations* refers to the dependence of nodes switching activity on the same nodes previous value. The *Spatial correlation* refers to the dependence of one nodes value on that of other nodes in the circuit and finally, when the nodes value depends on the other nodes previous values then its *Spatio-temporal correlation.* Many works have been done on power estimation with only input activity in to concern, but this assumption results in enormous errors in final power. Our model takes in to concern the input activity, spatial correlation and temporal correlation.

1.2 Domains of Description

As stated in earlier power estimation done at higher level of design specification are advantageous than compared to that done in the lower levels. To perform such a shift between different levels a detailed understanding of the different methods of design representation is vital. In [1] authors present three domains for circuit representation. The three domains of circuit representation are as follows:

- Behavioral Domain.
- Structural Domain.
- Physical Domain.

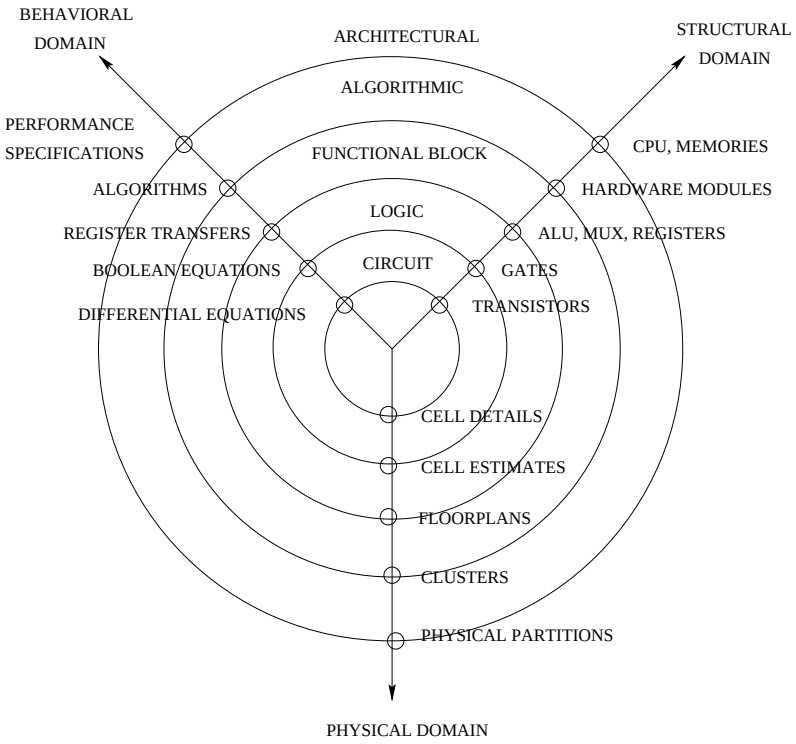

Figure 1.1. Design Representation

The behavioral domain specifies the behavior of the functionality of the design. At this stage only the static and dynamic components of the circuit are known. The static component refers to the time invariant portion of description namely particular operation - multiplication, subtraction and the dynamic component refers to the ordering of operations - sequencing, pipelining and timing. At this axes of domain specification, interconnection between the different components are not known. The logical structure of the design, the interconnection between different components are specified in this domain. The structural domain is the intermediate stage between the behavioral and physical domains. The physical domain of a circuit defines how each of the components described in their structural domain be actually implemented with real physical components. Speed, power and area constraints are part of the physical domain.

During the design specification, each of the three domains may also contain two new components namely, *description component* and the *constraint component.* The initial specifications of design which may be either specified by the user or may be the output from a control system forms

the *description component,* usually the description component refers to the input specification. The second component, *constraint component* refers to the various target requirements of the design, namely constraints on area, timing and power.

1.3 Levels of Abstraction

Domains of description specified in the previous section can be hierarchically decomposed in to different levels of abstraction. Figure 1.1. shows the the different domains along three axes and different levels of abstraction along the concentric circles. Each of these componenets are described in detail in the following section.

1.3.1 Architectural Level

Architectural level is the top most abstraction level often called as the system level. The behavioral domain at the architectural level specifies the behavior of the system, without considering the details on how the operation occurs. The components such as the control units, processors which perform such operations specified by the behavioral domain constitite the structural domain. As it the top most level of abstraction, more detailed logical structure is unavailable, and hence physical partitioning of the design details are specified by the physical domain.

1.3.2 Algorithmic Level

Algorithmic level also known as Behavioral level, describes the behavioral of the domain in terms of algorithms, flowcharts, processes and structures. The hardware modules that are used to represent the Behavioral domain, such as the control path and data path, is specified in the Structural domain. Clustering or partiotioning of similar operations that might be described in the structural domain are described in the Physical domain.

1.3.3 Functional Block Level

Also called as Register Transfer Level (RTL), this level forms the boundary between the logic gates and higher level representation. The arithmetic and logic operations of the data stored in regis-

ters constitute the behavioral domain at RT Level. Many methods are used to represent the behavior, namely the functional or dataflow method or through state machine oriented methods. Components suchas multiplexors, arithmetic logic units, comparators constitute the structural domain. They physical domain at this level deals with the floorplanning the layout of the design.

1.3.4 Logic Level

In Logic Level, also called as the Gate Level the behavior is specified by boolean equations. In the structural domain gates are used to implement the specified behavior. At this increasingly detailed specification level, the timing constraints may detailed with information on propagation delay, setup and hold times.

1.3.5 Circuit Level

This is the bottommost abstraction level with the behavior described in terms of differential equations representing the current and voltage terms, structural domain in terms of transistors, capacitors, diodes, resistors and the physical domain contributes to geometric specifications and their placement.

1.4 Components of Register Transfer Level

Register Transfer Level designs are composed of two components that interact with each other, namely the *datapath* and the *controller path.* The datapath consists of the execution units such as the adders, multipliers, multipliexors, buffers and registers. At higher level of design specification accurate power estimation is difficult due to the lack of sufficient implementation detail. The controller consists of a set of state machines and generates control lines for the datapath components.

1.5 Contributions of this Thesis

In this thesis we present a new method of power estimation that utilizes the behavioral description of the circuit and gate level implementation details to effectively model the power. When the behavioral specification is given and the gate level details are known, then the load capacitance can

be easily calculated. The switching activity at each node is then calculated using the Bayesian Networks. Models built based on independent inputs, random inputs are highly inefficient as when the input for the circuit is an output from another circuit the inputs are highly correlated. Our model works for both independent inputs and highly correlated inputs. Once the power is estimated using one type of implementation, another implementation of the same circuit can be estimated in no time and the best of the two circuits can be selected based on the design specifications. In this thesis work, two types of adder implemenations are tested for difference in power of the whole circuit. Though our model is zero delay model resulting in some reduction in estimation, our model is highly time efficient.

1.5.1 Behavior Induced DAG

The behavioral description is an Data Flow Graph (DFG) is used to construct a Bayesian Network(BIDAG), with nodes representing the variables and the arcs representating the dependencies between the variable. Once the bayesian network is constructed we use one of the stochastic sampling algorithm called the Evidence Pre-propagation algorithm to model the node dependencies. Sice the behavioral description includes components such as adders, multipliers, registers, subtractors, we identify each of the components inputs and feed it to another bayesian network (LIDAG) constructed using the components implementation details.

1.5.2 Logic Induced DAG

The terminal probabilities from the Behavioral Bayesian Network are fed to a bayesian constructed at the gate level. The Bayesian network constructed at a Logic level is called the Logic Induced Directed Acyclic Graph(LIDAG). Once the output probabilities are available from the BIDAG calculating power for any type of circuit implementation is very fast. We used HSPICE to model the load capacitances and to verify the outputs we used our own gate level simulator and the results are presented in the result section. This work focuses on mapping a Behavioral Data Flow Graphs (DFG's) in to Graphical Probabilistic networks called the Bayesian networks. We named our structure *BIDAG* as it is a combination of topdown and bottom up methodologies. We

prove that BIDAG is a bayesian network and then estimate power in RTL benchmark circuits to illustrate our case. Once the nodal switchings, which are inturn inputs to the internal components of the top-level circuit, are known, these probabilities are fed to an LIDAG,which is Logic Induced Directed Acyclic Graph that are used to represent the probabilistic knowledge of the gate level implementation. With this step all the internal node switching of the different components of the main circuit are extracted. The next step involves calculation of Power with capacitance extracted from gate level implementation. Since our method is a combinational top-down, bottom-up method, the capacitance values are already extracted. Average switching activity is estimated for the RTL components and used to calculate power. This power value is compared with gate level power estimate and results tabulated.

1.6 Flow of this Thesis

The outline of this thesis is as follows. Different power estimation techniques at Register Transfer level are discussed in the Chapter 2. In Chapter 3 we discuss the basic theory of Bayesian Networks and modelling a RTL circuit using bayesian network. After the construction of the Bayesian Network the inferencing, which is probabilistic updating, is explained in the Chapter 4 and finally we present our result in Chapter 5.

CHAPTER 2

RELATED WORK

Low power design has been the primary concern for current VLSI researchers. Shrinking the device sizes and packing millions of transistors in a single chip increases the power dissipation perunit area and this power has to be estimated in order to implement any low power technique. Power estimation has been performed at different levels of abstraction, such as the logic level, functional block level, algorithmic level and architectural level. Power estimation at logic level gives higher accuracy but is highly inefficient with time and cost as explained earlier. The power estimation at Register Transfer Level is faster than the gate level power estimation. Survey of different techniques of power estimation in Register Transfer Level has been best discussed in [2, 3].

The RTL power estimation can methods can be broadly categorized in to two methods, namely

- *top-down* methods.
- *bottom-up* methods.

Top-down methods are used when no information is available on the block whose power is to be estiamted. The block acts as a black box and the power is modelled with knowledge of the input statistics. These black box models are called as the *soft macros*. *Top-down* mothods are dependent on the initial conditions, such as the activity and the capacitance. If the internal details of the functional block is known then these method are highly inaccurate. *Bottom-up* methods are used when the internal implementation details of the functinal blocks are known. These functional blocks with synthesizable gate level specification of the hardware blocks are called the *hardmacros*. Figure 2.1. shows when bottom-up and top-down methods be applied. In the first case for the Adder, $A1$ no information is available on the type implimentation of adder. It might be a ripple carry adder,

carry propagate adder. But when implementation details are known it falls under to bottom-up methods.

2.1 Top-Down Power Estimation Techniques

Survey of various *topdown* techniques that operate at architectural, behavior, instruction and system domains has been done by Landman*et al.* [10]. In [5] Muller proposed a method to calculate power dissipation, area and speed based on the information from a knowledge base. The user has to decide on the design description like the estimated gate equivalents, cells, switching activity and capacitances and accuracy depends on the users judgement. These techniques have some accuracy in areas where complexity parameters are easily to estimate. In a similar technique based on the complexity [4], Liu*et al.* proposed differnt complexity parameters for different functional blocks and the switching activity factor was still assumed to be a user-specified constant. Entropy and Information theoretic mesaures were used to esitmate power in [6, 7] and [8, 9]. In [7] activity density of the functional block, D is estimated based on the entropy. Given the probability of the signal is p then, the entropy of the node is given by,

$$H(x) = plog_2\frac{1}{p} + (1-p)log_2\frac{1}{1-p} \tag{2.1}$$

If the signal can take n variables then the entropy is given by,

$$H(x) = \sum_{i=1}^{n} p_i log_2\frac{1}{p_i} \tag{2.2}$$

Using Equation 2.2 input and output entropies, H_{inp}, H_{out}, are estimated and the average entropy, H_{avg}, of the functional block is calculated using the equation 2.3.

$$H_{avg} \approx \frac{2/3}{n+m}(H_{inp} + 2H_{inp}) \tag{2.3}$$

Entropy is good representation of the signal activity when the signal is considered independent$p =$ 0.5. Hence these methods are best suited when the random inputs are considered at the inputs. In [14] power estimation for soft macros, functional block for which synthesizable HDL is available

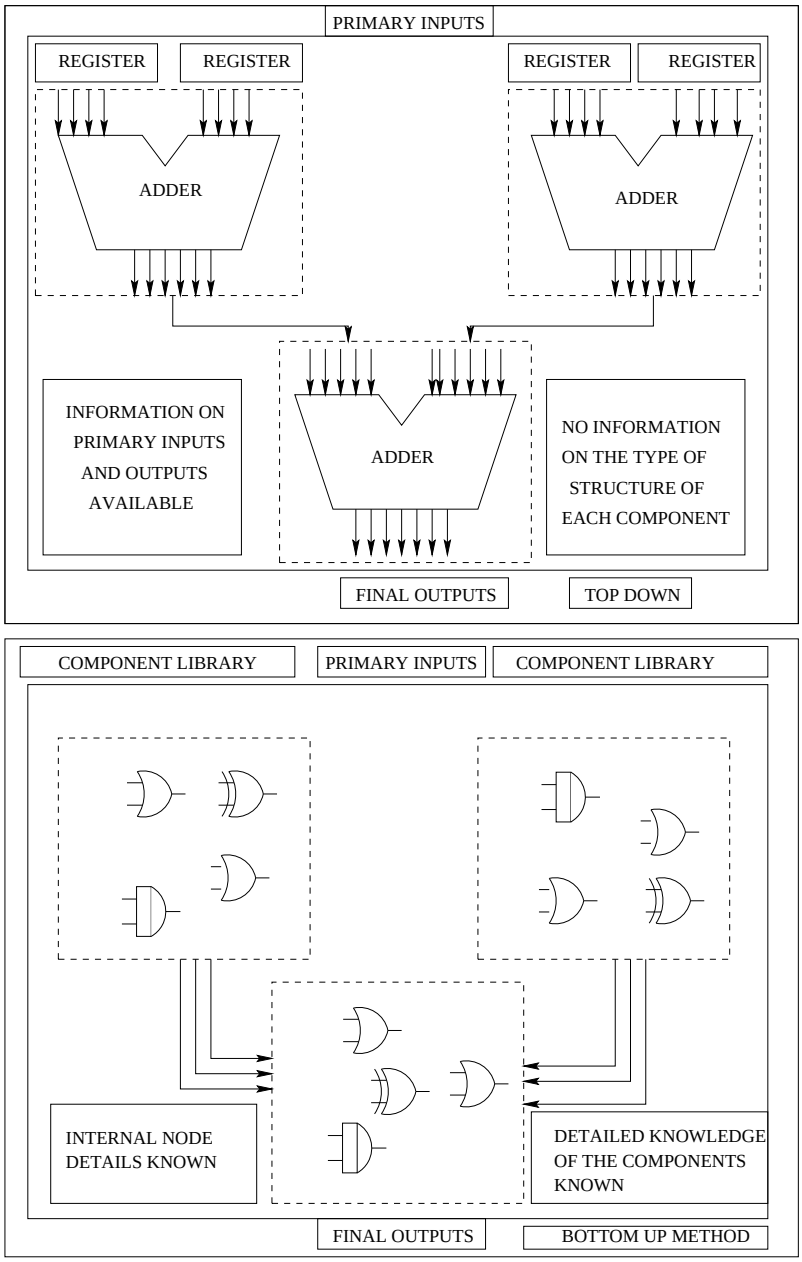

Figure 2.1. Top-Down and Bottom-Up Methods

but the gate level implementations, depending on the input and output activity is proposed. Characterization is based on a technique of adaptive signal processing known as *least mean squares*.

2.2 Bottom-Up Power Estimation Techniques

Several *bottom-up* power estimation techniques have been proposed recently. When the implementation details are known, bottom-up techniques have more accuracy than the *top-down* methods. Though extensive research has been in the estimation of average power, works on cycle-by-cycle power has also been presented. Most *bottom-up* methods are based on *macromodeling*. A power macromodel is constructed for a functional block and is then used for high level power estimation.

Macromodels are classified in to two types :

- *equation-based macromodels.*
- *look-up table based macromodel.*

The simplest power macromodeling technique proposed in [15] called the *power factor approximation* models the average power with the assumption that the inputs are highly independent. This assumption results in loss of accuracy as the power is dependent on input correlations. In [16] slightly improved approach was proposed by Landman *et al.*. The inputs are not completely independent, but the temporal correlations are considered in the MSBs and the LSBs are considered random. This assumption can be more accurate if more statistics like the spatial correlations and spatio-temporal correlations were considered. More works on regression based models include [17, 18, 21]. Since regression is dependent on the training pattern, accuracy is decreased when operated under different input patterns then the ones used to characterize the macromodel. In [18] two macromodeling techniques were proposed for time effective and more accurate macromodeling, namely the *Sampler macromodeling* and *Adaptive macromodeling.* In *Sampler macromodeling* technique inputs are sampled based on a random sampling method there by reducing the number of cycles in which the input statistics are studied thereby reducing the time for simulation. *Adaptive macromodeling* is based on regression analysis utilizing gate level power simulation and hence record better accuracy.

One of the pioneering work by Najm *et al.* [21] used Look-Up-Table based characterization approach to to build the power macromodel and has been shown to have imporved accuracy and robustness. This work by Najm *et al.* is an extension of the work by the same author in [36]. In [36] a three dimensional table is constructed whose axes are *average input probability* (P_{inp}), *average input transition density* (D_{inp}) and *average output zero delay transition density* (D_{out}). For any selection of the three values results in a power value for the module. But this has more error since two values of (P_{inp}), (D_{inp}), (D_{out}) does not always result in same power. Hence Najm*et al.* introduced a fourth dimension in to the table, *Input Spatial Correlations* SC_{inp}. In the previous sentences *input probability,* is the probability that the signal is '1', *transition density,* is the number transitions per unit time, *spatial Correlation* refers to the dependence of the particular node on any node. The average power based on the four dimensional lookup table is given by,

$$P_{avg} = f(P_{inp}, D_{inp}, SC_{inp}, D_{out}) \tag{2.4}$$

Each of the variables that constitute the axes of the macromodel satisfy the condition given below,

$$\frac{D_{inp}}{2} \leq P_{inp} \leq 1 - \frac{D_{inp}}{2} \tag{2.5}$$

$$\frac{nD_{inp}^2 - P_{inp}}{n-1} \leq SC_{inp} \leq P_{inp} \tag{2.6}$$

Similar Look-up-tables are utilized in power macromodeling as in [20, 34, 22, 23]. Input dependent and input independent models depend on characterization. Characterization is the process that improves accuracy of power models by exploiting accurate simulations of the gate level implementation of the unit to be modeled, repeated for significant input transitions. Applications such as noise analysis, heat dissipation analysis need power to be calculated at every cycle. Clutering approach proposed [34] to calculate the cycle-by-cycle power is based on the assumption that closely related input transitions have similar power dissipation. This assumption is not always true and when the number of clusters becomes large it results in inaccurate results. An automatic procedure

for cycle-accurate macromodel generation was proposed by Wu *et al.* [19]. Improved characterization of macromodels are addressed in [11, 20, 29, 28, 27, 30, 31]. [20] is a Look-up-table based approach, where inputs are clustered for higher accuracy. The average power is then an given by,

$$P_{avg} = f(P^L_{inp}, P^H_{inp}, D^L_{inp}, D^H_{inp}, D_{out}) \tag{2.7}$$

P^L_{inp} is the probability of the low switching input, P^H_{inp} is the probability of high switching input, D^L_{inp} density of the low switching input, D^H_{inp} is the density of the high switching , D_{out} is the density of the output. Clustering the inputs based on the switching in to high and low switching increases the dimension of the Lookup Table and there by increases the characterization time. Power model using node sampling was proposed in [29]. The assumption that power can be modeled with partial knowledge of few nodes is highly prone to errors. Another node sampling technique was proposed in [27] where instead of internal nodes are sampled instead of the input nodes. Another cycle power estimation technique proposed by Potlapally *et al.* in [11] is based on seperating the input space in to regions with similar power behavior and seperate macromodels are constructed for each of these input spaces. The selection of the appropriate power model for a given space is determined by a function called *Power mode Identification Function* (PIF). Power macromodeling based on power sensitivity was proposed in [33]. Power is represented as a function of power sensitivity to primary input activity $\zeta_a(x_i)$ given by Equation 2.8 and power sensitivity to primary input probability $\zeta_P(x_i)$ given by Equation 2.9. The final power is cum of power due to normal average power dissipation P_{norm} and sensitivity and is given by the Equation 2.10. Spurious activity in Register Transfer Level is addressed in [24, 25].

$$\zeta_a(x_i) = \frac{\delta Power_{avg}}{\delta a(x_i)} = \sum_{i \in primaryinputs} fanout(j) \frac{\delta a(j)}{\delta a(x_i)} \tag{2.8}$$

$$\zeta_a(x_i) = \frac{\delta Power_{avg}}{\delta P(x_i)} = \sum_{i \in primaryinputs} fanout(j) \frac{\delta a(j)}{\delta P(x_i)} \tag{2.9}$$

$$Power = Power_{nom} + \sum_{i \in primaryinputs} (\zeta_a(x_i)) \triangle a(x_i) + (\zeta_{Px_i}) \triangle P(x_i) \tag{2.10}$$

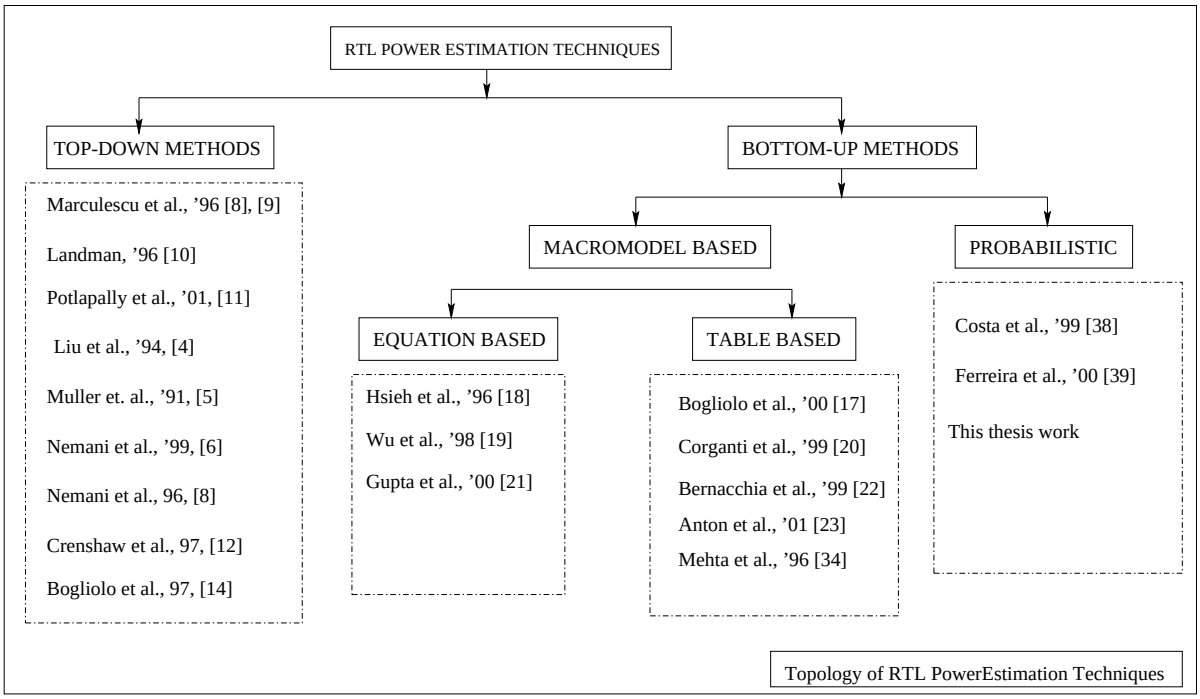

Figure 2.2. Register Transfer Level Power Estimation Techniques

CHAPTER 3

BAYESIAN NETWORKS FOR REGISTER TRANSFER LEVEL POWER MODELING

To solve a complex problem one needs to have a clear knowledge of the nature of the problem and the reason why it had happened. In most of the real life situations the knowledge of the problem is obscure and coming to conculsions based on these uncertain knowledge is highly erraneous, thus requiring a different method of representation. Bayesian networks, also called as Bayesian belief networks, casual networks or probabilistic networks, are graphical models, that utilize the probabilistic and statistical techniques, to concisely represent the knowledge of the uncertainity.

To illustrate better about the need for Bayesian networks, consider circuit of a full adder shown in Figure 3.1.. The probability distribution, that gives the current knowledge or *belief,* of the curcuit or also called as the *belief* for the adder, is a function with every input, output and present state of the gate and is represented as,

$$P(A, B, C, I1, I2, I3, I4, I5, I6, Sum, Carry) \tag{3.1}$$

The real time intelligent system domains are huge with more number of variables in the probabilistic distribution. Increase in number of variables eventually makes the belief updating untractable. The graphical probabilistic models on the other hand utilize the dependency between the nodes and the resulting probability distribution is simpler. Bayesian network for the full adder is shown in the Figure 3.2., where each of the nodes represent the random variables and the links between the nodes determine the influence of one node over other. Here nodes *A*, *B*, *C* are input nodes, $I1$, $I2$, $I3$, $I4$ are internal nodes and *Sum*, *Carry* are output nodes. When the belief of the nodes $I1$ and $I4$ are known then the *posterior belief* of *Sum* can be updated with knowledge of its parent, $I1$ and $I4$. This independency of a child node over all other nodes, given the its parent nodes

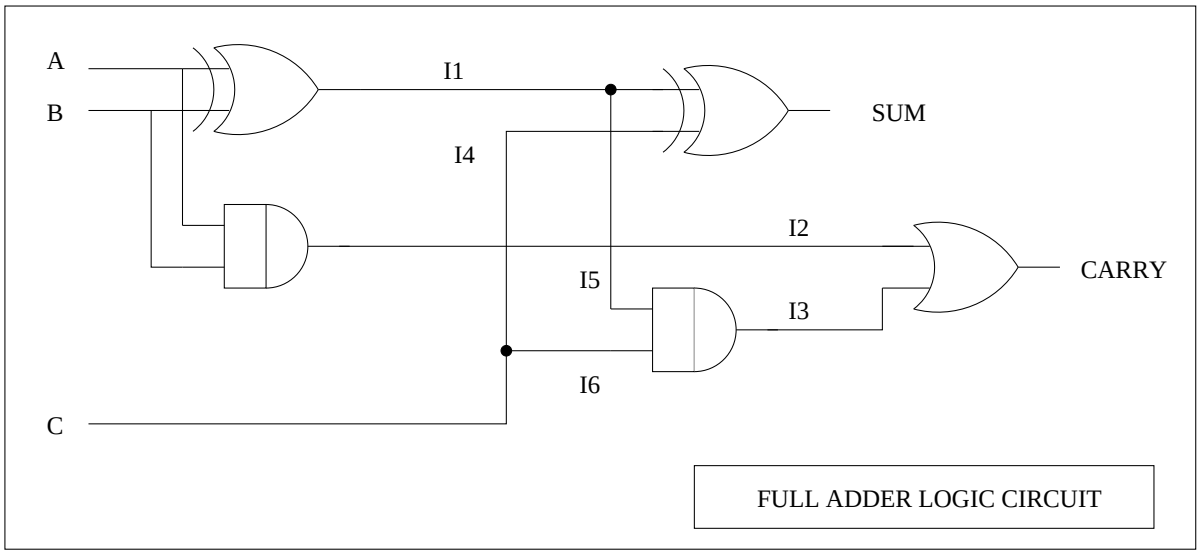

Figure 3.1. Full Adder Logic Circuit

belief is called as the conditional independence. A Bayesian network represents the exponentially sized joint probability distribution in a compact manner. The joint probability distribution for a net with *n* variables, with belief of parent $Pa(X_k)$,is given by,

$$P(X_1, \cdots, X_N) = \prod_{k=1}^{n} P(X_k | Pa(X_k) \tag{3.2}$$

Hence graphical models that utilize the conditional independence, are used to concisely represent the probabilistic knowledge of the circuit. The domain refers to a collection of variables in a set. Each of these variables have a discrete space, from where they get their values. The term *Belief updating* refers to updating the knowledge of the prior belief with posterior belief. The graphs that are used to represent the bayesian network and update the current belief of a network can be of three types, namely *directed*, *undirected* and *hybrid.* The nodes in the graph represent the variable and the edge between the nodes represent the direct dependence between the nodes. As the name suggests, in undirected graphs as shown in Figure 3.3.(*b*) the nodes are connected by an undirected edge and the path represented by $< N1, N5, N7, N8 >$. The directed graph is shown in the Figure 3.3.(*a*) with the edge between the nodes replaced with directed arcs. The graph is directed acyclic graph and

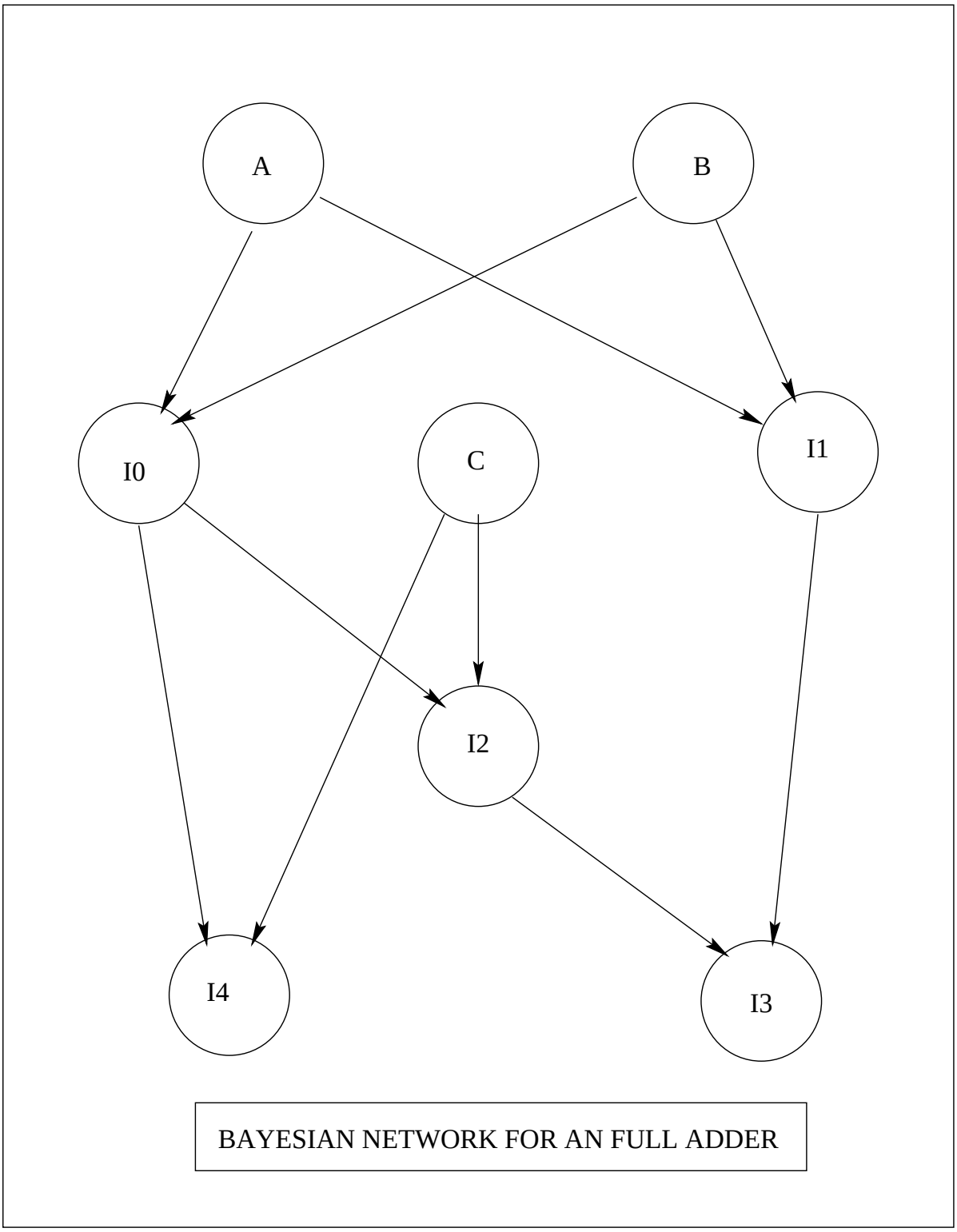

Figure 3.2. Bayesian Network for the Full Adder

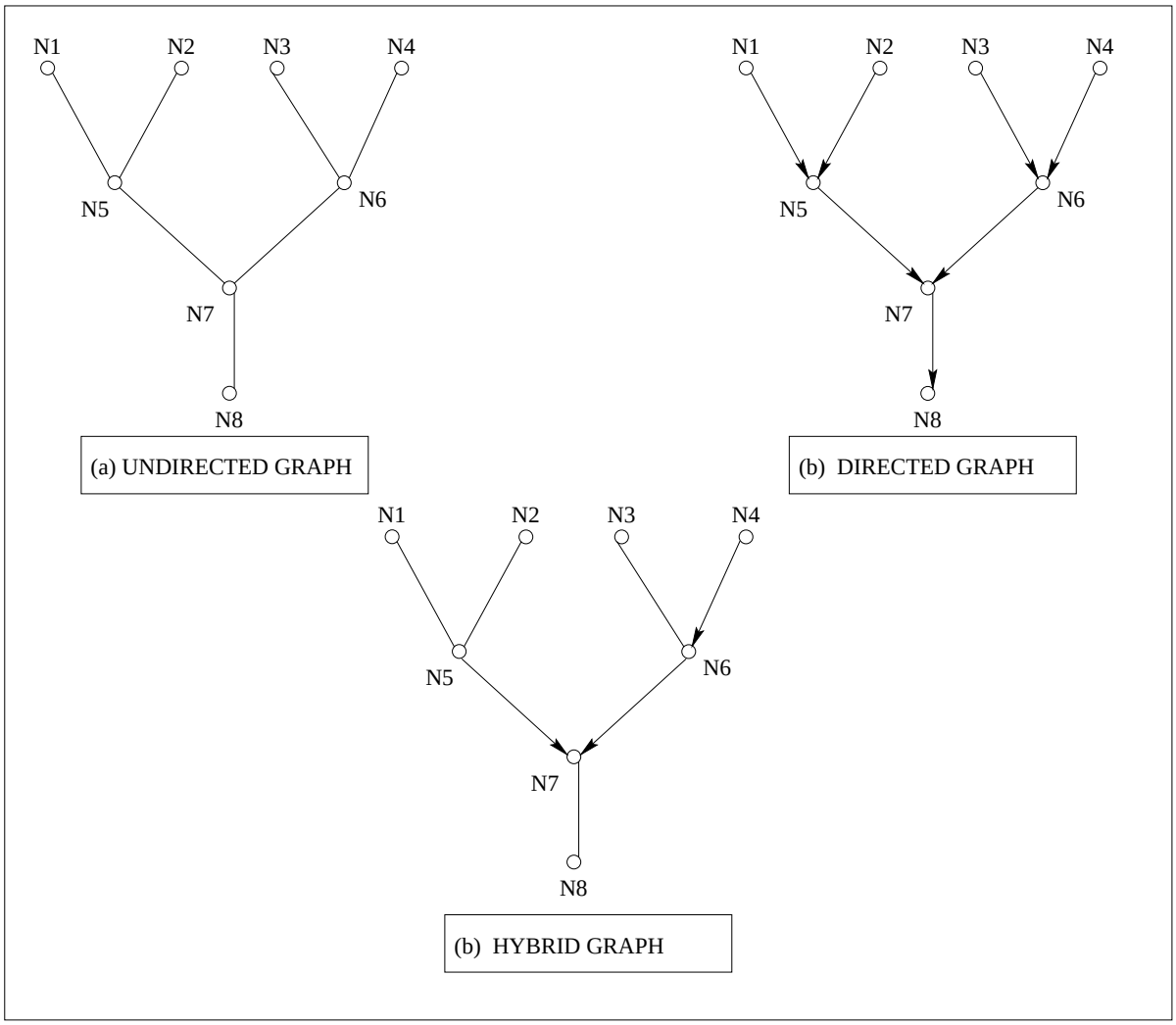

Figure 3.3. Graphical Models used to Represent a Digital Circuit

the path is represeted as $< N4, N6, N7, N8, N10 >$ and every node has a incoming head and has an outgoing head of the arrow. A hybrid graph is combination of cyclic and acyclic graphs.

3.1 Bayesian Networks

Bayesian network is a graphical model that efficiently encodes the joint probability distribution for large set of variables. Each variable v_i gets its value from a finite space D_{v_i}. In our case in order to model the switching activity at the nodes of the bayesian network, the finite space for each of the variables is four in length, each containing knowledge on switching namely, $0 \rightarrow 0, 0 \rightarrow 1, 1 \rightarrow$

$0,1 \rightarrow 1$. The design space is then the cartesian product of the spaces for the variables. Each of the element in D_{v_i}, is called a configuration. The uncertainities in the dependence between variables are represented in terms of a probability function P defined over the variables. $P(X = x|Y = y)$ represents the belief on the truth of x given the truth of y and is termed as conditional probability and unconditional distribution over the entire domain is referred to as /joint probability distribution.

In the following section we will discuss various notations and definitions from Pearl [44].

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

$$P(x|y,z) = P(x|z) \text{ whenever } P(y,z) > 0 \tag{3.3}$$

$I(X,Z,Y)$ represents the conditional independence of X and Y, which states that X and Y are independent on each other when the knowledge of the other subset Z is known. In the Figure 3.3. having a knowledge of the node, $I1$, makes the variables A and *Sum* independent of each other. A dependency model, M, of a domain should capture all these triplets namely $(A, I1, Sum)$. 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:

$$I(X,Z,Y) \Rightarrow I(Y,Z,X) \quad \text{(symmetry)} \tag{3.4}$$

$$I(X,Z,Y \cup W) \Rightarrow I(X,Z,Y) \& (X,Z,W) \quad \text{(decomposition)} \tag{3.5}$$

$$I(X,Z,Y \cup W) \Rightarrow I(X,Z \cup W,Y) \quad \text{(weak union)} \tag{3.6}$$

$$I(X,Z,Y)\&I(X,Z\cup Y,W) \Rightarrow I(X,Z,Y\cup W) \quad \text{(contraction)} \tag{3.7}$$

Symmetry axiom states that if an observation of Z is available and if X has no influence on Y, then Y has no information on X. The decomposition axiom states that if two sets are irrelevant to X having a knowledge of Z, then they are individually irrelevant. The axiom of weak union suggests that learning an irrelevant information W cannot make Y more relevant of X. If W is found irrelevant to X after learning some irrelevant information Y, then W was irrelevant to X even before we learned Y.

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 above two conditions hold, the path is called an active path.

In the Figure 3.3., consider the nodes *Sum*, $I4$ and $I1$. Node *Sum* is connected with B via $I2$ and $I4$. There exists two intermediate nodes in between the primary node and the product node. The node $I2$ *d-seperates* the nodes $I4$ and B, which inturn is deseperated from node *Sum* by the $I2$. $I4$ and $I1$ acting as direct parents to the *Sum* node cannot be further *d-seperated.* Next we discuss about the DAG with with such d-seperations with conditional independence.

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 the Figure 3.3. nodes *Sum*, $I4$, *Is* exhibit conditional independence, when node *Sum* is unchanged no matter by the presence of absence of the knowledge of $I2$, which is shown in the equation 3.8. Further seperation of the nodes results in loss of dependency and information and thus this minimized d-seperated graphs form the minimal I-map and the DAG with minimum I-maps is called as a Bayesian Network. This is illustrated in the following definitions.

$$p(Sum|A,B,C,I1,I2,I4) = p(Sum|I1,I4) \tag{3.8}$$

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.

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.

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) have to be ordered before the random variable.

Theorem 2: Let M be any dependency model satisfying the axioms of independence listed in Eqs. 3.4-3.7. If the graph structure D is a boundary DAG of M relative to ordering d, then D is a minimal I-map of M.

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.

Definition 8: At a Behavioral level, the Directed Acyclic Graph (DAG) represents the behavior of the system, where each node represents the variables, including primary, intermediate and output, and edges specify the direct depedency of the nodes being connected. The intermediate nodes are the result of outputs from multipliers, adders or comparators. Each node carry the switching probability information which is then traversed to the child nodes to which the out going directed graphs from primary node are connected. The DAG for the behavioral specifiation, called the

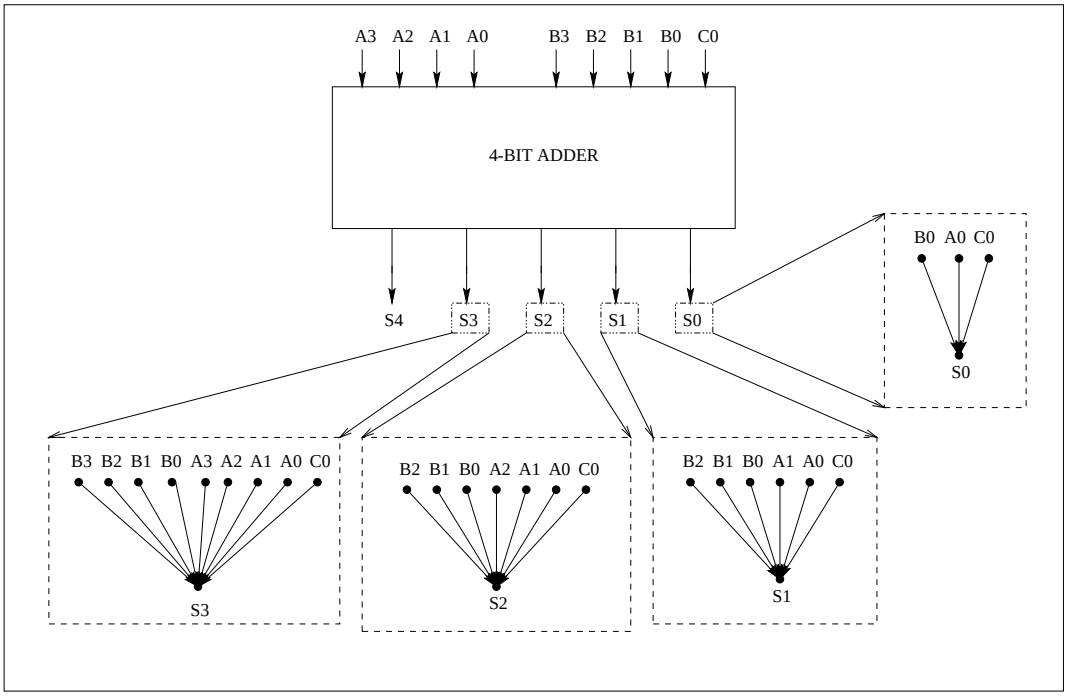

Figure 3.4. Parents for Each of the Output Bits of an 4-Bit Adder

Behavior Induced Directed Acyclic Graph (BIDAG) is shown for an IIR Filter Figure 3.7.. Here the nodes $A0$, $A1$, $A2$, $A3$, $A4$, $B0$, $B1$, $B2$, $B3$, $B4$, $B5$ are the primary nodes, $I1$, $I2$, $I3$, $I4$, $I5$, $I6$, $I7$, $I8$, are the intermediate nodes as a result of addition or multiplication and *Yout* is the output node.

The dependencies of the output bits of an 4-Bit Adder is shown in Figure 3.4.. It is evident from the figure that as the size of the Adder increases, the net number of parents for each of the output bit also increases and thereby the joint probability distribution for any higher order output bit is a complex function. In the example shown in Figure 3.4., number of parents for the S_3 bit is 9. In order to overcome the complex functions that result from the previous structure, dummy nodes are introduced in the circuit to minimize the size of the probability distribution. These dummy nodes effectively capture the dependence relation between a node to its parents. Introduction of dummy nodes in 4-Bit Adder circuit is shown in Figure 3.5. where nodes, $C1$, $C2$, $C3$, $C4$ are the dummy nodes that contribute to a case where any output node is dependent only on a maximum of 3 inputs.

Theorem 3: The BIDAG structure is a minimal I-map of the underlying switching dependency model and hence is a Bayesian network. *Proof:* To better understand, let us take the data flow graph

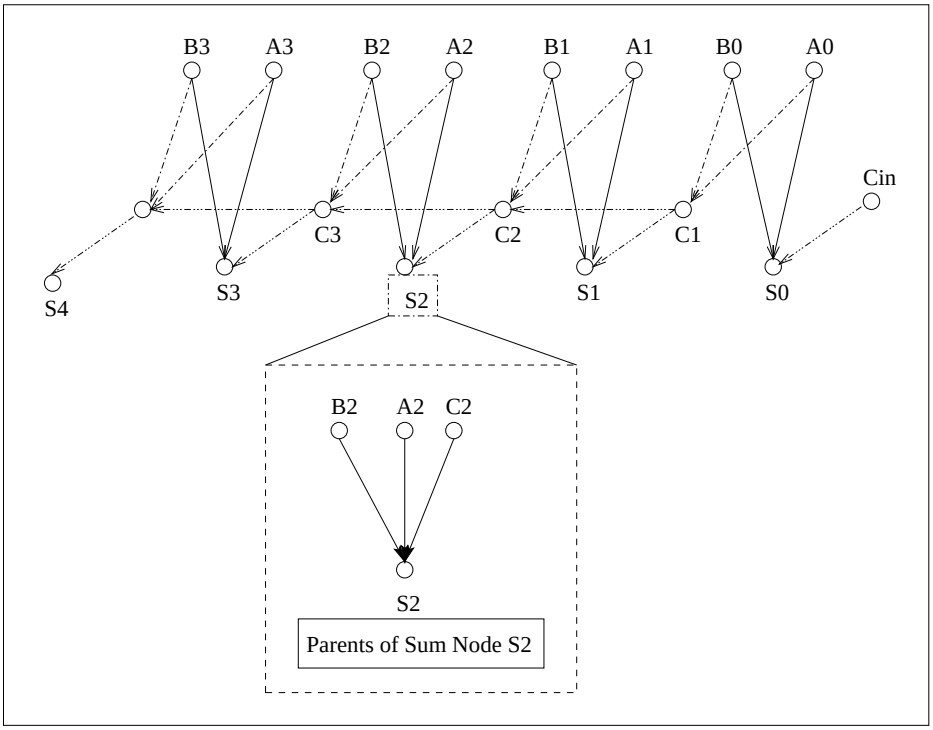

Figure 3.5. Parents for Each of the Output Bits of an 4-Bit Adder after Introduction of Dummy Nodes

of the fulladder shown in 3.5., where nodes, A, B represent set of all the bit inputs to the adder and the bits C, S denote the carry or dummy nodes and the sum nodes. It is clear from the figure that *Sum* nodes are dependent onthe previous bit adder output *Carry*, which in turn is conditionally dependent on the primary inputs at that previous stage. Though $A0, B0, A1, B1, C1, C2$ constitute the distant and immediate parent of the node $S2$, knowledge of switching at the intermediate node which form its immediate parent or called as the *Markov boundary*, $C2$ is sufficient enough to model the *Sum* and *Carry*, thus proving that $I(S2|C2|A1)$ and any switching in the input nodes will have causal effect on the output nodes, in our case, Adders. Thus formed BIDAG is a Directed Acyclic Graph structure corresponds exactly to the DAG structure one would arrive by considering principles of causality which states that one can arrive at a appropriate Bayesian Network by directing links from nodes that are causes to nodes that represent immediate effects, with the directed link the immediate causes of switching that quality.

3.2 Behavior Induced Directed Acyclic Graph

The power modeling presented in this thesis work utilizes information from two levels of design specification, namely *gatelevel* and $RegisterTransferLevel$, thus two different types of bayesian networks are required to be constructed. The construction of such bayesian networks, BIDAGs for Behavioral level, and Gate level LIDAGs are explained in the following section.

At the behavioral level, the circuit is specified in terms of registers, arithmetic units, such as adders, multipliers and a typical Data Flow Graph (DFG) is suitable enough to represent the circuit, as shown for a IIR filter in Figure. 3.6.. This graph can be transformed in to a bayesian network with nodes acting as the variables and the arithmetic functions, addition, subtraction, multiplication, be represented by a *directed arc*. This directed arc captures all the dependencies of the child node, eg. $I1$ multiplier output, on the parent node $A0$. Each variable or node in the network holds the probability of each type of transitions $\{A0_{00}, A0_{01}, A0_{10}, A0_{11}\}$. Four bit inputs are considered and hence each $A0$ is subdivided in to $A00$, $A01$, $A02$, $A03$ and are independent of each other unless they are outputs from another component that might decide the input of the current circuit in hand, the

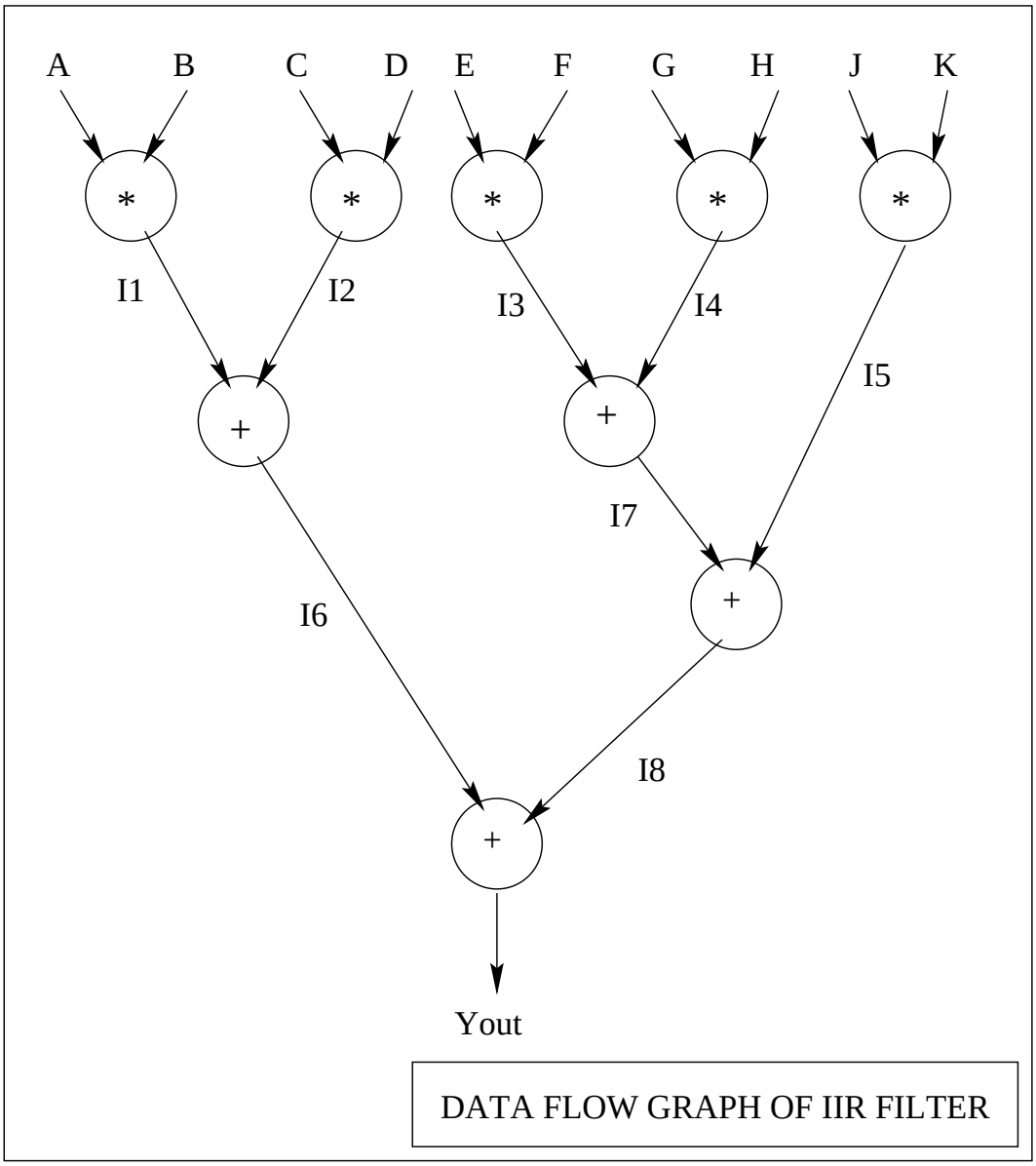

Figure 3.6. Data Flow Graph for an IIR Filter

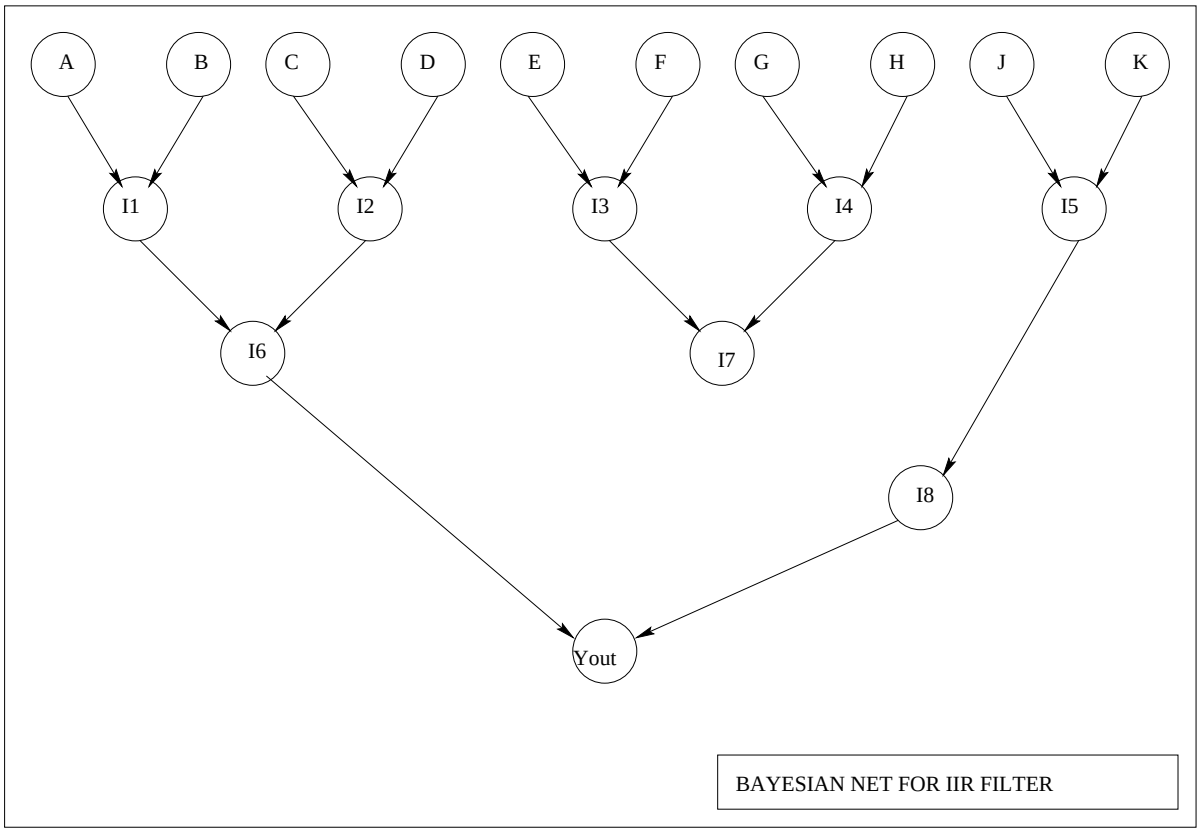

Figure 3.7. Directed Acyclic Graph for an IIR Filter

IIR filter. The flow of messages, for belief updating discussed in the next chapter, depends on the conditional probability table constructed for the each of the units, multipliers, adders.

In the case of an adder, which determines nodes $I7$, $I8$ and $Yout$, behavioral format of the adder's output sum and carry, the equation format, is utilized to construct the Conditional Probability Table. To make it more clear, the adder *Sum* and *Carry* equations are given by Equations. 3.9 and 3.10.

$$Sum_{i-1} = (A_{i-1} + B_{i-1} + Carry_{i-1})\%2 \tag{3.9}$$

$$Carry_i = \frac{((A_{i-1} + B_{i-1} + Carry_{i-1}) - Sum_i)}{2} \tag{3.10}$$

Switching in the *Sum* and *Carry* is studied by observing the previous values of the *Sum* and *Carry*. The Conditional Probability table for the *Sum* of the Adder structure is shown in the Ta-

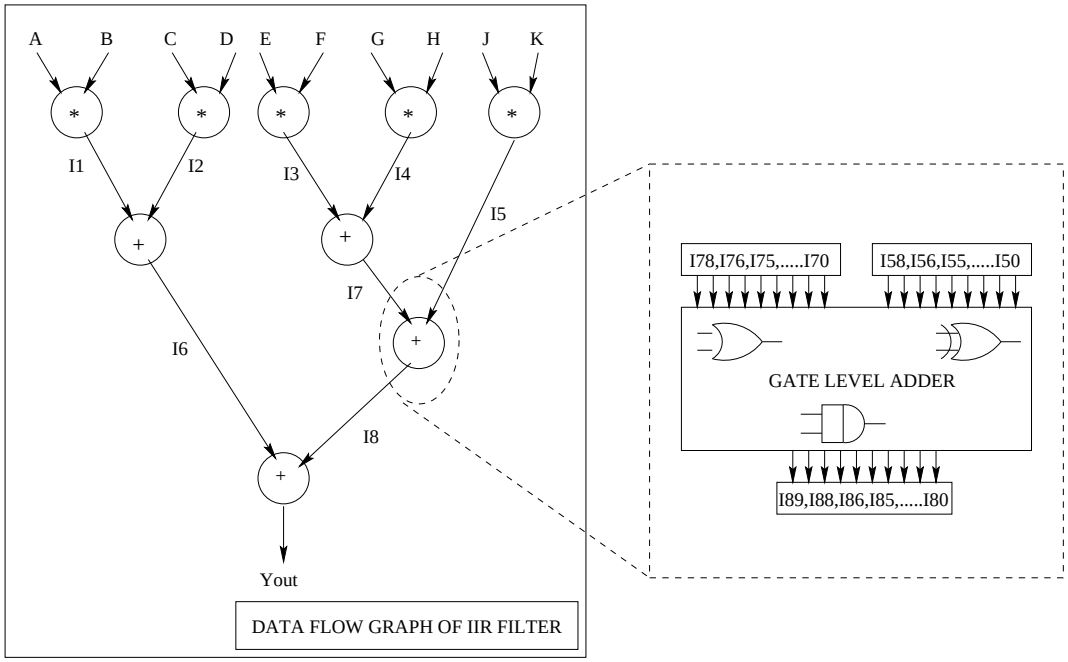

Figure 3.8. Terminal Node Switching Propagated to the LIDAG

bles. 3.1.. Here X_{00}, X_{01}, X_{10}, X_{11} represent switching from $0 \rightarrow 1$, $0 \rightarrow 1$, $1 \rightarrow 0$, $1 \rightarrow 1$ for inputs X_{input1}, X_{input2}, X_{input3} and output X_{output}. Conditional Probabilty tables of such kind are built for each of the intermediate nodes and a full Bayesian network is constructed with all the nodes of the circuit. This bayesian network with the information on the input node probability determines the output probabilites of each intermediate nodes effectively.

3.3 Logic Induced Directed Acyclic Graph

Since our method is sandwitch method with knowledge of both higher level specification and lower level implementation details, in order to compute power logic level bayesian network (LIDAG) is constructed with the knowledge of switching from the Behavioral level Bayesian Network (BIDAG). This is best illustrated in the Figure. 3.8.. Once the belief updating is done for the behavioral bayesian network, terminal probabilities of the different components are fed to the gate level bayesian network. The terminal probabilities of the output of adder and multiplier, $I7$ and $I5$ are fed to the gate level Bayesian Network, LIDAG. Since this freedom of design selection is available, these

Table 3.1. Conditinal Probability Table for the Sum Bit of an Adder

$P(X_{output}\|X_{input1},X_{input2})$ for $X_{output}=$				X_{input1}	X_{input2}	X_{input3}	$P(X_{output}\|X_{input1},X_{input2})$ for $X_{output}=$				X_{input1}	X_{input2}	X_{input3}
$\{x_{00}$	x_{01}	x_{10}	$x_{11}\}$	=	=	=	$\{x_{00}$	x_{01}	x_{10}	$x_{11}\}$	=	=	=
1	0	0	0	x_0	x_0	x_0	0	0	1	0	x_2	x_0	x_0
0	1	0	0	x_0	x_0	x_1	0	0	0	1	x_2	x_0	x_1
0	0	1	0	x_0	x_0	x_2	1	0	0	0	x_2	x_0	x_2
0	0	0	1	x_0	x_0	x_3	0	1	0	0	x_2	x_0	x_3
0	1	0	0	x_0	x_1	x_0	0	0	0	1	x_2	x_1	x_0
1	0	0	0	x_0	x_1	x_1	0	0	1	0	x_2	x_1	x_1
0	0	0	1	x_0	x_1	x_2	0	1	0	0	x_2	x_1	x_2
0	0	1	0	x_0	x_1	x_3	1	0	0	0	x_2	x_1	x_3
0	0	1	0	x_0	x_2	x_0	1	0	0	0	x_2	x_2	x_0
0	0	0	1	x_0	x_2	x_1	0	1	0	0	x_2	x_2	x_1
1	0	0	0	x_0	x_2	x_2	0	0	1	0	x_2	x_2	x_2
0	1	0	0	x_0	x_2	x_3	0	0	0	1	x_2	x_2	x_3
0	0	0	1	x_0	x_3	x_0	0	1	0	0	x_2	x_3	x_0
0	0	1	0	x_0	x_3	x_1	1	0	0	0	x_2	x_3	x_1
0	1	0	0	x_0	x_3	x_2	0	0	0	1	x_2	x_3	x_2
1	0	0	0	x_0	x_3	x_3	0	0	1	0	x_2	x_3	x_3
0	1	0	0	x_1	x_0	x_0	0	0	0	1	x_3	x_0	x_0
1	0	0	0	x_1	x_0	x_1	0	0	1	0	x_3	x_0	x_1
0	0	0	1	x_1	x_0	x_2	0	1	0	0	x_3	x_0	x_2
0	0	1	0	x_1	x_0	x_3	1	0	0	0	x_3	x_0	x_3
1	0	0	0	x_1	x_1	x_0	0	0	1	0	x_3	x_1	x_0
0	1	0	0	x_1	x_1	x_1	0	0	0	1	x_3	x_1	x_1
0	0	1	0	x_1	x_1	x_2	1	0	0	0	x_3	x_1	x_2
0	0	0	1	x_1	x_1	x_3	0	1	0	0	x_3	x_1	x_3
0	0	0	1	x_1	x_2	x_0	0	1	0	0	x_3	x_2	x_0
0	0	1	0	x_1	x_2	x_1	1	0	0	0	x_3	x_2	x_1
0	1	0	0	x_1	x_2	x_2	0	0	0	1	x_3	x_2	x_2
1	0	0	0	x_1	x_2	x_3	0	0	1	0	x_3	x_2	x_3
0	0	1	0	x_1	x_3	x_0	1	0	0	0	x_3	x_3	x_0
0	0	0	1	x_1	x_3	x_1	0	1	0	0	x_3	x_3	x_1
1	0	0	0	x_1	x_3	x_2	0	0	1	0	x_3	x_3	x_2
0	1	0	0	x_1	x_3	x_3	0	0	0	1	x_3	x_3	x_3

Table 3.2. Conditional Probability Specifications for Output and Input Line Transitions for Two Input AND Gate

Two Input AND gate					
$P(X_{output}\|X_{input1}, X_{input2})$ for $X_{output} =$ $\{x_{00}$	x_{01}	x_{10}	$x_{11}\}$	X_{input1} =	X_{input2} =
1	0	0	0	x_{00}	x_{00}
1	0	0	0	x_{00}	x_{01}
1	0	0	0	x_{00}	x_{10}
1	0	0	0	x_{00}	x_{11}
1	0	0	0	x_{01}	x_{00}
0	1	0	0	x_{01}	x_{01}
1	0	0	0	x_{01}	x_{10}
0	1	0	0	x_{01}	x_{11}
1	0	0	0	x_{10}	x_{00}
1	0	0	0	x_{10}	x_{01}
0	0	1	0	x_{10}	x_{10}
0	0	1	0	x_{10}	x_{11}
1	0	0	0	x_{11}	x_{00}
0	1	0	0	x_{11}	x_{01}
0	0	1	0	x_{11}	x_{10}
0	0	0	1	x_{11}	x_{11}

terminal behavior form the BIDAG can be fed to any type of the gate level circuit implementation. Gate level bayesian network is a Directed Acyclic Graph with nodes representing the variables and the edges giving the dependencies between the connected nodes. At the gate level the edges hold the conditional probability information of the particular gate type, eg. XOR gate, NAND gate. The conditional probability table for an AND gate that is used in the full adder is shown in the Table. 3.2.. Here $\{X_{00}, X_{01}, X_{10}, X_{11}\}$ represent switching from $\{0 \rightarrow 1, 0 \rightarrow 1, 1 \rightarrow 0, 1 \rightarrow 1\}$ for inputs $\{X_{input1}, X_{input2}$ and output X_{output}. As the gate level specifications are known in this method, switching activity at each node is calculated with the knowledge of capacitance from the component library. Thus constructing a different implementation of the same function, eg. addition, can be tested for power and the best of the two implementations can that meets user specifications can be used in the final model. We used two implementations of adders, namely ripplecarry adder and carry propagate adder, to address the shift in implementations and difference in power.

CHAPTER 4

BAYESIAN NETWORK INFERENCE

Inference in probabilistic expert systems refers to calculating probability distribution of a query node given an observation or evidence. The two main tasks of bayesian network inference are *belief updating* and *belief revision. Belief updating* is the process of calculating the posterior probability of an query node, X, given the observed value of the evidence node, E, which is given by the equation. 4.1

$$P(X|E=e) = \frac{P(X/E, E=e)}{P(E=e)} \tag{4.1}$$

Belief revision refers to the most probable instantiation of the variables, given the observed evidence. When the variables that are to tbe computed are non evidence nodes, then the method is also called as computing a most probable explanation, MPE. Most probable explanation refers to computing most probable explanation.

Bayesian Influence algorithms are broadly classified in to two types, namely, *Exact Inference* and *Approximate Inference*. Exact inference algorithms are best suited for small circuits and Approximate inference algorithms are used when belief updating is to be done for large circuits with higher number of nodes.

4.1 Exact Inference Algorithms

Exact Inference algorithms are best suited for small circuits with less number of loops, also called a the *cliques.* Number of exact inference algorithms were proposed. Perl proposed exact inference algorithms for message passing poly trees, one of which is loop cutset conditioning where the connectivity of a network is changed by an algorithm and a subset of nodes called loop cutset

are formed. Polytree algorithm is used to solve these singly connected loop cutsets. As different instantiations have to be considered while forming the loop cutsets these methods are highly complex and fail for higher number of nodes. Clique-tree propagation is another common bayesian network inference algorithm, where a multiply connected network is converted in tot clique tree by clustering the triangulated moral graph of the undirected graph. The clique tree algorithm also fails for complex networks. Other methods of exact inference algorithms include the *conditioning, arc reversal*, *Symbolic probabilistic infrence, elimination* and *differential* methods. In the Arc reversal/node reduction method, links between different nodes are inversed using Bayes rule to such a state that the evidence nodes are directly connected to the query nodes. In the differential method partial derevatives of multivariate polynomial of the bayesian network is used to compute the probabilistic queries of the hypothesis nodes. Bayesian network inference algorithms discussed so far can be applied to networks with less number of nodes and that are simpler. When the complexity of the network increases and with the increase in the node size, belief updating by exact inference is impossible and hence we go for another set of inference algorithms called Approximate Inference Algorithms.

4.2 Approximate Inference

To meet with the NP-hard nature of *belief updating* in dense networks with large node sizes, approximate algorithms are used to get the probability of an query node for a given evidence. Approximate inference algorithms work by generating random samples for the variables using an pseudo random number generator and approximating the conditional probabilities of the query nodes. Since the probabilities of nodes converge to a value only when almost all the possible combinations are captured by the given sample, these methods are effective when given a long samples and when time is not a constraint. Input samples are randomly generated and hence they are independent on each other. Approximate Inference algorithms also called as *Stochastic Simulation Algorithms*, are broadly classified in to two types, namely *Importance Sampling Algorithms* and *Markov Chain Monte Carlo* methods.

Stochastic Simulation Algorithms are broadly classfied in to the following two types:

- *Forward Sampling*
- *Backward Sampling*

In the *Forward sampling* stochastic simulation algorithms pseudo random numbers are generated and propagated from the input nodes traversed by the topological order whereas *Backward sampling* stochastic simulation algorithms start from the evidence nodes and find the best fit for rest of the node probabilites. Forward sampling techniques can be *Probabilistic Logic Sampling, Likelihood weighing* and Backward sampling techniques include Importance sampling algorithms, such as *Adaptive Importance Sampling, Evidence Pre Propagation Importance Sampling,* to generate samples based on an importance function. Backward Sampling algorithms are mainly useful when there is unlikely evidence nodes when forward sampling techniques are highly inaccurate in calculating the posterior probabilities.

4.2.1 Probabilistic Logic Sampling

The first and the most simple method of stochastic inferencing is the probabilistic logic sampling [52]. Given evidence for certain nodes, a random number generator and traversed to trough the topology of the network to the child node. During each run, the probabilites at each node is captured and the average of the node probability is calculated at the end of the sampling. When evidence nodes are present, then the samples during which the inconsistent probabilities of evidence node values are rejected and simulation is done for matching evidence node probabilites. Probabilistic Logic Sampling is highly erraneous when the evidence is very unlikely, when the number of samples which satisfy the set evidence is very few than the actual number of samples need for the probabilities to converge. Probabilistic Logic Sampling techniques are suited for small circuits with very few or no evidence nodes.

4.2.2 Likelihood Weighting

Problem caused due to the Probabilistic Logic Sampling is handled in addressed in the Likelihood weighting algorithm [54]. Unlike the previous method where the samples are rejected for inconsistent evidence values, the observed value of the evidence variable are used to calculate weight

of the sample called as the *score* for the probability of an event. Score models the fraction of probability of an event in the sample that matches with the evidence to the probability of the same event considering every sample no matter it matches evidence or not.

Importance sampling algorithms exhibit improved sampling approach by using an importance sampling function, to approximate the posterior probability distribution. To compute Integral as shown in Equation. 4.2, similar to computing the probability of a node, importance sampling approach is given in Equation. 4.3. The function $f(X)$ is the importance function

$$I = \int_{\Theta} g(X)dX \tag{4.2}$$

$$\hat{(}I) = \int_{\Theta} \frac{g(X)}{f(X)} f(X)dX \tag{4.3}$$

Samples that are used to approximate the conditional probability are dependent on the function $f(X)$. Once the sampling is done the integral function is given by,

$$\hat{(}I) = \int_{\Theta} \frac{g(S_i)}{f_{S_{s_i}}} f(X)dX \tag{4.4}$$

The variance of the probability table is inversely proportional to increase in the number of nodes. The importance algorithms can be classified based on two classes, namely, *Sel sampling algorithm* and *heuristic importance sampling.* Circuit importance function is updated, using the scores generated in the algorithm, to revise the conditional probability tables in order to make the sampling algorithm approach to the estimaton. In the heuristic importance sampling method edges are removed in the network to make it similar to polytrees and then a poly tree algorithm is used to compute the likelihood functions. If the heuristic importance function evaluaved is close enough to the optimal importance then it can lead to a significant improvement in performance.

4.2.3 Adaptive Importance Sampling

Cheng*et al.* in [50] proposed an importance sampling algorithm called the Adaptive Importance Sampling, (AIS), that minimizes the sampling variance by varying the importance function so that it

is close enough to the optimal importance function. The optimal importance sampling function, that is a complex mathematical expression, for calculating the posterior probabilities given the evidence of query nodes is identical to the importance sampling function when same network structure is used during the evidence. Then optimal importance which captures effect of all evidence on every node in the network is given by Equation 4.5

$$\rho(X/E) = \prod_{k=1}^{m} P(X_k | Pa(X_k, E)) \tag{4.5}$$

Importance Conditional Probability Tables (ICPT) are used to update the posterior probabilities of each node conditional on the evidence nodes. Instead of incrementing the occurance as in the previous sampling techniques, ICPT tables very similar to the Conditional probabilitly tables are updated during the importance function learning process.

The third main reason for improved results in Adaptive Importance Sampling araises from the initialization of the importance function. Two heuristics are presented for better performance. While learning the importance function, if the initial value of the importance function is close to the optimal importance function, convergence is achieved at an earlier time. The immediate ancestral nodes to the evidence nodes are the most affected nodes than ones that are further down the path. Cheng*et al.* had proposed that by initializing the Importance Conditional Probability Tables of immediate ancestral nodes higher convergence rates are achieved. Addressing uncertain evidence problem in the previous techniques, he had suggested that if a threshold Θ can be set such that when any node probability is below the threshold value it is replaced with the threshold value. In turn this threshold is deducted from the largest probabiliy in the same conditional probability distribution.

4.2.4 Evidence Pre Propagated Importance Sampling

Proposed by Yuan *et al.* in [51], Evidence Pre-propagated Importance sampling uses local message passing and stochastic sampling techniques to effectively calculate the probabilities of query nodes for given evidence nodes. EPIS is similar to the Adaptive Importance sampling method except that the learning to learn the approximations of the ICPTs are direct. If $X = \{X_1, X_2, \cdots, X_i\}$ is a set of variables in Bayesian network, let $Pa(X_i)$ be the parent of Xi and E be the set of evidence.

The optimal importance function is then given by the Equation 4.6, where $P(X_i|Pa(X_i,E))$ is defined as the importance conditional probability table.

$$P(X|E) = \prod_{k=1}^{n} P(X_i|Pa(X_i,E)) \tag{4.6}$$

Definition : An importance conditional probabililty table on X_i is a table of posterior probabilities $P(X_i|Pa(X_i,E))$ conditional on the evidence and indexed by it immediate predecessors, $Pa(X_i)$.

Every node in the poly tree d-seperates in to two subsets called the E^- and E^+. E^- denotes the evidence connected to the children and E^+ denotes the evidence connected to the parent of any node X_i. The two subsets are independent on each other and the evidence message thus passed are termed as, $\lambda(x)$ messages for those sent by the parents and $\pi(x)$ messages for those sent from the children to the parents.

$$\lambda(x) = P(X_i|Pa(X|E^-)) \tag{4.7}$$

$$\pi(x) = P(X_i|Pa(X|E^+)) \tag{4.8}$$

After the messages are propagated and convergence is achieved the posterior belief on any node is given by,

$$Bel(x) = \alpha\lambda(x)\pi(x) \tag{4.9}$$

Perls belief propagation algorithm can be applied to networks with loops where the belief of the node is continuously updated in a loop till belief converges. Similar to the AIS method, in EPIS method the threshold for the low probability in the network is identified and replaced with an θ threshold and at the same time the largest probability of the network is subtracted by this cutoff.

Stochastic sampling techniques discussed above work fine for bayesian networks as the optimal importance function is the product of the conditional probability function of all nodes. We used EPIS algorithms to estimate the switching activty for our circuits, and we found that samples as low as 5000 are sufficient enough for all the circuits. In addition to the input, the input state spaces are

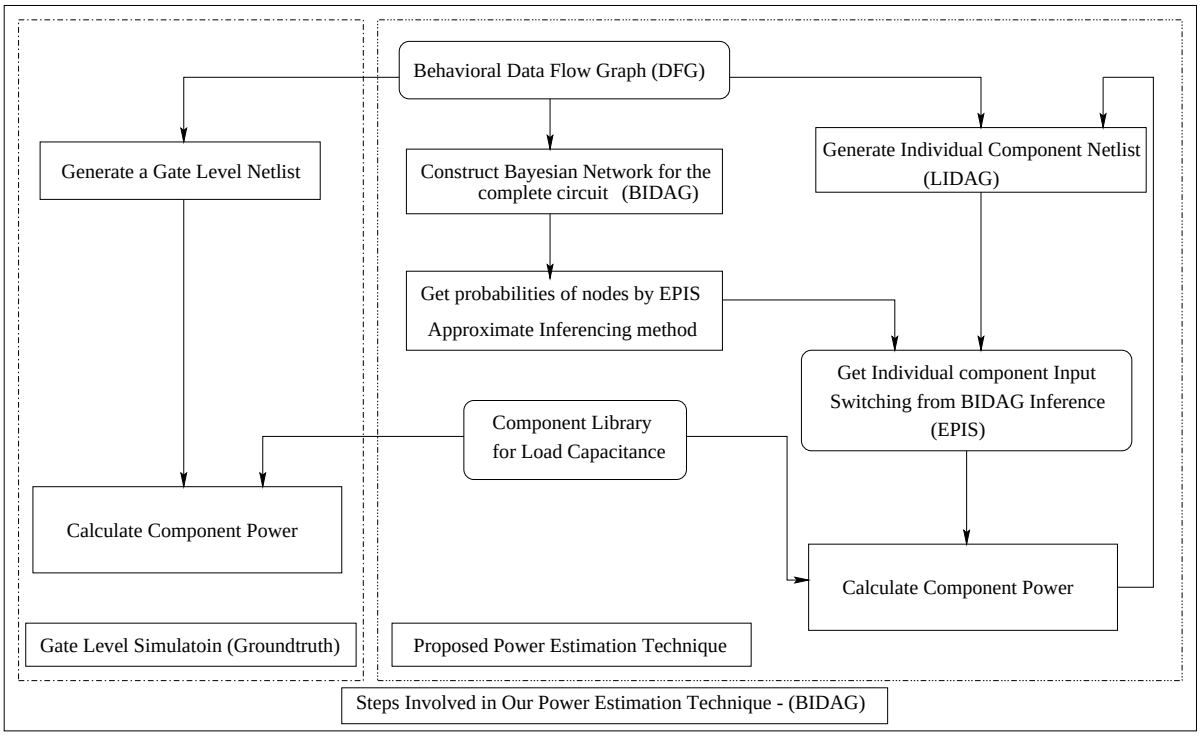

Figure 4.1. Power Estimation Steps

sampled simultaneously using a strong corrective model as captured by Bayesian network. Another important advantage of this method is that it is input pattern insensitive and the convergence rate is very high.

CHAPTER 5

EXPERIMENTAL RESULTS

In this section we discuss the power estimates at RT level benchmark circuits. For every benchmark, we convert them into a data-flow graph whose nodes are the resources like adder, multiplier and edges are inputs and outputs of the respective resources. We then translate this DFG to a Behavioral-DAG (BIDAG) which is proven to a Bayesian Networks which the minimal I-map for the underlying logical dependencey inherent in the algorithm. The estimation steps are highlighted in Figure 4.1.. Once we obtain the BIDAG, we infer the probabilities of the inputs/outputs using Evidence Pre-propagated Importance Sampling which models the uncertainty in the system and the probabilities that we obtain match the belief of the system.

Next, we use logic level implementation of individual resources and model them as LiDAGs. We use the inferred probabilities from the BIDAG to the corresponding LiDAG structure and use the same belief propagation to obtain the switching profile of the individual internal signals, the power dissipation of the entire system is thus computed. Thus, we partition our problem into handling logic level dependency using the BIDAG structure and handle the structural dependency using the LIDAG structure.

We use WINDOWS XP computer with Pentium IV, 2.00GHz processor to run our Inference algorithms and Sun Solaris machines for our gate level simulations. We tested our model for three

Table 5.1. Results on Total Dynamic Power Dissipation for Benchmark Circuits

	Low Input Switching $\{x_{00} = 0.5x_{01} = 0.15x_{10} = 0.15x_{11} = 0.2\}$			
Circuit	*No.of Nodes*	*EPIS*	*Simultion*	*||%error||*
FIR	1573	0.00143	0.00143	0.00
IIR	1522	0.00141	0.001411	0.00
ELLIPTIC	9081	0.0069	0.0070	1.4

Table 5.2. Total Dynamic Power Dissipation for Benchmark Circuits

	Random Input Switching $\{x_{00} = 0.25 x_{01} = 0.25 x_{10} = 0.25 x_{11} = 0.25\}$			
Circuit	*No.of Nodes*	*EPIS*	*Simultion*	*||%error||*
FIR	1573	0.002438	0.00244	0.08
IIR	1522	0.002430	0.002428	0.08
ELLIPTIC	9081	0.001	0.0103	2.90

Table 5.3. Total Dynamic Power Dissipation for Benchmark Circuits

	High Input Switching $\{x_{00} = 0.1 x_{01} = 0.4 x_{10} = 0.4 x_{11} = 0.1\}$			
Circuit	*No.of Nodes*	*EPIS*	*Simultion*	*||%error||*
FIR	1573	0.002811	0.002811	0.00
IIR	1522	0.002798	0.002798	0.00
ELLIPTIC	9081	0.01154	0.01177	1.90

types of input behavior and for ELLIPTIC filter which have reconvergence paths we have a maximum error of only 3%. The first sample is an for low input switching behavior, shown in Table 5.1. where as circuit power for high input switching behavior is shown in Table 5.3.. The completely random input behavior which is the condition in non feed back circuits is discussed in Table 5.2.. Thus our model has less then 3% and is input pattern independent. In Table 5.4. we have presented the power for the filters that result due to a different type of adder, Carry Propagate Adder under three input switching behavior. Finally we have discussed the closeness of our BiDAG switching estimates with the gate level switching estimates. Table 5.5., 5.6., 5.7. shows the average output node switching in each component in the FIR filter, table. 5.8., 5.9., 5.10. shows the average output node switching in each component in the IIR filter and Table 5.11., 5.12., 5.13. shows the average output node switching in each component in the ELLIPTIC filter. We compared this switching with gate level switching estimate and the maximum error of 2% was recorded for the 4 bit multiplier.

Table 5.4. Results on Total Dynamic Power Dissipation for Benchmark Circuits for Different Implementaiton of Adder

	Power for Different Input Switching			
Circuit	No. of Nodes	*SampleI*	*Sample*2	*Sample*3
FIR	4388	0.0027	0.0041	0.0046
IIR	3769	0.0025	0.0039	0.0042
ELLIPTIC	27939	0.0131	0.0184	0.0201

Table 5.5. Results on BIDAG Switching vs Actual Switching from Simulation for FIR Filter

	Sample I		
Modlue	*Simulation*	*EPIS*	‖%*error*‖
4MULT	0.254	0.25645	0.9
8ADD	0.334	0.3347	0.2
9ADD	0.3498	0.3510	0.3
10ADD	0.3767	0.3802	0.9
11ADD	0.3339	0.3291	1.4

Table 5.6. Results on BIDAG Switching vs Actual Switching from Simulation for FIR Filter

	Sample II		
Modlue	*Simulation*	*EPIS*	‖%*error*‖
4MULT	0.4206	0.4254	1.1
8ADD	0.4432	0.4407	0.05
9ADD	0.4280	0.4248	0.07
10ADD	0.4208	0.4178	0.07
11ADD	0.3808	0.3809	0.02

Table 5.7. Results on BIDAG Switching vs Actual Switching from Simulation for FIR Filter

	Sample III		
Modlue	*Simulation*	*EPIS*	‖%*error*‖
4MULT	0.4532	0.4534	0.00
8ADD	0.4580	0.4583	0.06
9ADD	0.4340	0.4313	0.62
10ADD	0.4173	0.4142	0.74
11ADD	0.3884	0.3866	0.46

Table 5.8. Results on BIDAG Switching vs Actual Switching from Simulation for IIR Filter

	Sample I		
Module	*Simulation*	*EPIS*	‖%*error*‖
4MULT	0.2629	0.2575	2.0
8ADD	0.3344	0.3351	0.2
9ADD	0.3555	0.351	1.2
10ADD	0.3682	0.3641	1.1

Table 5.9. Results on BIDAG Switching vs Actual Switching from Simulation for IIR Filter

	Sample II		
Module	*Simulation*	*EPIS*	$\|\|\%error\|\|$
4MULT	0.4202	0.4206	0.09
8ADD	0.4436	0.4428	0.1
9ADD	0.4279	0.4255	0.5
10ADD	0.4155	0.4172	0.4

Table 5.10. Results on BIDAG Switching vs Actual Switching from Simulation for IIR Filter

	Sample III		
Module	*Simulation*	*EPIS*	$\|\|\%error\|\|$
4MULT	0.4532	0.4519	0.28
8ADD	0.4581	0.4579	0.04
9ADD	0.4340	0.4356	0.36
10ADD	0.4239	0.4256	0.39

Table 5.11. Results on BIDAG Switching vs Actual Switching from Simulation for ELLIPTIC Filter

	Sample I		
Module	*Simulation*	*EPIS*	$\|\|\%error\|\|$
4SUB	0.4179	0.4245	1.5
4MULT	0.2635	0.2664	1.08
8ADD	0.3005	0.3019	0.46
4ADD	0.4091	0.4081	0.24
9SUB	0.3191	0.3211	0.62
8SUB	0.3212	0.3208	0.12
9MULT	0.1865	0.1843	1.19
8MULT	0.209	0.2093	0.14
18ADD	0.2200	0.219	0.45
19ADD	0.2204	0.2192	0.54

Table 5.12. Results on BIDAG Switching vs Actual Switching from Simulation for ELLIPTIC Filter

	Sample II		
Module	*Simulation*	*EPIS*	$\|\|\%error\|\|$
4SUB	0.4716	0.4735	0.40
4MULT	0.4001	0.4137	3.2
8ADD	0.3807	0.3933	3.2
4ADD	0.4989	0.4997	0.16
9SUB	0.3949	0.3936	0.33
8SUB	0.3876	0.3886	0.27
9MULT	0.2629	0.2628	0.03
8MULT	0.2716	0.2721	0.18
18ADD	0.2756	0.2762	0.21
19ADD	0.2673	0.2669	0.14

Table 5.13. Results on BIDAG Switching vs Actual Switching from Simulation for ELLIPTIC Filter

	Sample III		
Module	*Simulation*	*EPIS*	‖%*error*‖
4SUB	0.4434	0.4507	1.6
4MULT	0.4297	0.4443	3.3
8ADD	0.3929	0.4059	3.3
4ADD	0.5310	0.5318	0.15
9SUB	0.3873	0.3997	3.2
8SUB	0.3936	0.3955	0.4
9MULT	0.2781	0.2770	0.39
8MULT	0.2850	0.2855	0.17
18ADD	0.2848	0.2842	0.21
19ADD	0.2735	0.2719	0.58

CHAPTER 6

CONCLUSION

Power estimation has been an active research topic for more than a decade. This thesis introduces combinational Top-down Bottom-up method to accruately model the power at Register Transfer Level. We have shown the results of Power for various RTL behchmark circuits. We used Bayesian networks to get the probability of any query node given the evidence for any other node. We used an Importance sampling called Evidence Pre-propagation Importance Sampling method to update the belief. Since 5000 samples resulted in good convergence in the probabilities, it is much faster than power macromodeling where the table construction take most of the time. Our probelm is zero delay model and hence the extension of our work would be real dealy model for RTL power estimation.

REFERENCES

[1] R. A. Walker and D. E. Thomas, "A Model of Design Representation and Synthesis," *IEEE/22nd Design Automation Conference*, pp. 453– 459, 1985.

[2] F. N. Najm, "A Survey of Power Estimation Techniques in VLSI circuits", *IEEE Transactions on VLSI Systems*, vol. 2, No. 4, pp. 446-455, 1994.

[3] E. Macii, M. Pedram and F. Somenzi, "High-Level Power Modeling, Estimation and Optimization", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1061-1079, vol. 17, no. 11, Nov. 1998.

[4] D. Liu and C.Svensson, "Power Consumption Estimation in CMOS VLSI Chips", *IEEE Journal of Solid State Circuits*, pp.663-670, 1994.

[5] K. Muller-Glaser, K. Kirsch and K. Neusinger, "Estimating Essential Design Characteristics to Support Project Planning for ASIC Design Management", *IEEE International Conference on Computer Aided Design*, pp.148-151, 1991.

[6] M. Nemani and F. N. Najm, " High-Level Area and Power Estimation for VLSI Circuits", *IEEE Transactions on computer-Aided Design of Integrated Circuits and Systems*, vol. 18-6, pp. 697-713, June 1999.

[7] M. Nemani and F. N. Najm, "Towards a High-Level Power Estimation Capability [Digital ICs]", *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, vol.15-6, pp. 588-598, June 1996.

[8] D. Marculescu, R. Marculescu and M. Pedram, "Information Theoretic Measures for Power Analysis", *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems (Special Issue on Low Power Design)*, vol.15-6, June 1996.

[9] D. Marculescu, R. Marculescu and M. Pedram, "Theoretical Bounds for Switching Activity Analysis in Finite-State Machines", *IEEE Trans. on VLSI Systems (Special Issue on Low Power Design)*, vol.8-3 ,June 2000.

[10] P. Landman, "High-level power estimation",*ACM/IEEE Int. Symp. Low-Power Electronics and Design*, pp.29-35, 1996.

[11] N. R. Potlapally, A. Raghunathan, G. Lakshminarayana, M. S. Hsiao, S. T. Chakradhar, "Accurate Power Macro-modeling Techniques for Complex RTL Circuits", *IEEE VLSI Design Conference*, pp. 235-241, Jan. 2001.

[12] J. E. Crenshaw and M. Sarrafzadeh, "Accurate High Level Datapath Power Estimation", 1997.

[13] R. Zafalon, M. Rossello, E. MAcii, M. Poncino, “Power MAcromodeling for a High Quality RT-level Power Estimation”, *IEEE First International Symposium on Quality Electronic Design*, pp.59-63, 2000.

[14] A. Bogliolo, L. Benini and G. D. Micheli, “Adaptive Least Mean Square Behavioral Power Modeling”, *Eur. Design and Test Conference*, pp. 404-410, 1997.

[15] S. Powell and P. Chau, “Estimating Power Dissipation of VLSI Signal Processing Chips: The PFA Technique”, *VLSI Signal Processing*, pp.250-259, 1990.

[16] P. E. Landman and J. M. Rabaey, “Architectural Power Analysis: The Dual Bit Type Method”, *IEEE Transactions on Very Large Scale Integration(VLSI) Systems*, vol.3-2, pp. 173-187, June 1995.

[17] A. Bogliolo, L. Benini and G. D. Micheli,”Regression-Based RTL Power Modeling”, *ACM Transactions on Design Automation of Electronic Systems*, vol. 5-3, pp.337-372, July 2000.

[18] C. Hsieh, Qing Wu, C. Ding and M. Pedram, ”Statistical Sampling and Regression Analysis for RT-Level Power Evaluation”, Proceedings of the International Conference on Computer Aided Design, pp.583-588, Nov. 1996.

[19] Q. Wu, Q. Qiu, M. Pedram and C-H. Ding,”Cycle-Accurate Macro-Models for RT-Level Power Analysis”, *IEEE Trans. on VLSI*, vol.6-4, pp. 520-528, December, 1998.

[20] R. Corgnati, E. Macii and M. Poncino, “ Clustered Table-Based Macromodels for RTL Power Estimation”, *Proceedings Ninth Great Lakes Symposium on VLSI*, pp. 354-357, March 1999.

[21] S. Gupta and F N. Najm, “ Analytical Models for RTL Power Estimation of Combinational and Sequential Circuits”, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol.19-7, pp. 808-814, Jul 2000.

[22] G. Bernacchia, M. C. Papaefthymiou, “Analytical Macromodeling for High-Level Power Estimation”, *IEEE/ACM International Conference on Computer-Aided Design*, pp. 280-283, Nov. 1999.

[23] M. Anton, I. Colonescu, E. Macii and M. Poncino, “ Fast Characterization of RTL Power Macromodels”, *IEEE International Conference on Electronics, Circuits and Systems*, vol. 3, pp. 1591-1594, Sep. 2001.

[24] A. Raghunathan, S. Dey and N. K. Jha, “ Register-Transfer Level Estimation Techniques for Switching Activity and Power Consumption ”, *IEEE/ACM International Conference on Computer-Aided Design, ICCAD*, pp. 158-165, Nov. 1996.

[25] A. Raghunathan, S. Dey and N. K. Jha, “Glitch Analysis and Reduction in Register Transfer Level Power Optimization”, *33rd Design Automation Conference Proceedings*, pp. 331-336, june 1996.

[26] D. Bruni, G. Olivieri, A. Bogliolo and L. Benini, “ Delay-Sensitive Power Estimation at the Register-Transfer Level”, *IEEE International Conference on Electronics, Circuits and Systems*, vol. 2, pp. 1031-1034, Sep. 2001.

[27] A. Bogliolo and L. Benini, “Robust RTL Power Macromodels”, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 6, pp. 578-581, no. 4, Dec. 1998.

[28] A. Bogliolo, L. Benini, G. De. Macheli, “Characterization-free behavioral power modeling”, *Proceedings of Design, Automation and Test in Europe*, pp. 767-773, Feb. 1998.

[29] A. Bogliolo and L. Benini, “Node Sampling: A Robust RTL Power Modeling Approach”, *IEEE/ACM International Conference on Computer-Aided Design, ICCAD*, pp. 461-467, Nov. 1998.

[30] M. Eiermann, W. Stechele, “Novel Modeling Technique for RTL Power Estimation”, *Proceedings of the 2002 International Symposium on Low Power Electronics and Design*, pp. 323-328, Aug. 2002.

[31] M. Eiermann, W. Stechele, “RTL Power Modeling Techniques for Combinational and Sequential RTL macroblocks”, *International Conference on Electronics, Circuits and Systems*, vol. 2, pp. 705-508, Sept. 2002.

[32] Yi-Min Jiang, Shi-Yu Huang, Kwang-Ting Cheng, D. C. Wang, Ching and Yen Ho,“A Hybrid Power Model for RTL Power Estimation”, *Proceedings of the ASP-DAC, Design Automation Conference*, pp. 551-556, Feb. 1998.

[33] Z. Chen and K. Roy, ”A Power Macromodeling Technique Based on Power Sensitivity”, *ACM/IEEE Design Automation Conference*, pp. 678-683, 1998.

[34] H. Mehta, R.M. Owens and M.J. Irwin, “Energy Characterization Based on Clustering”, *Proceedings of ACM/IEEE Design Automation Conference*, pp.702-707, 1996.

[35] S. Gupta and F. N. Najm, “Energy-Per-Cycle Estimation at RTL”, *International Symposium on Low Power Electronics and Design*, pp. 121-126, Aug. 1999.

[36] S. Gupta and F. N. Najm, “Power Macromodeling for High Level Power Estimatoin”, *Proceedings of 34th Design and Automation Conference*, pp. 365-370, Anaheim, California, June 9-13, 1997.

[37] T. Sato, Y. Ootaguro, M. Nagamatsu and H. Tago, “Evaluation of Architecture-Level Power Estimation for CMOS RISC Processors”, *Symposium on Low Power Electronics*, pp.44-45, 1995.

[38] J. Costa, J. Monteiro, L. M. Silveira and S. Devadas, “A Probabilistic Approach for RT-Level powre modeling”, *The 6th IEEE International Conference on Electronics, Circuits and Systems*, September 1999.

[39] R. Ferreira, A. M. Trullemans, J. Costa and J. Monteiro, “Probabilistic Bottom-Up RTL Power Estimation”, *Proceedings of IEEE First International Symposium on Quality Electronic Design* , pp. 439-446, Mar. 2000.

[40] S. Bhanja and N. Ranganathan, “Dependency preserving probabilistic modeling of switching activity using Bayesian networks,” *IEEE/ACM Design Automation Conference*, pp. 209–214, 2001.

[41] S. Bhanja and N. Ranganathan, “Switching Activity Estimation of VLSI Circuits Using Bayesian Networks,” *IEEE Transactions on VLSI Systems*, vol. 11, no. 4, pp. 558–567, Aug. 2003.

[42] S. Bhanja and N. Ranganathan, ”Switching activity estimation of large circuits using multiple Bayesian networks,” *Proceedings of ASP-DAC and 15th International Conference on VLSI Design*, pp. 187–192, 2002.

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

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

[45] “Graphical Network Interface”

`URL http://www.sis.pitt.edu/~genie/genie2.`

[46] S. M. Kang, “Accurate Simulation of Power Dissipation in VLSI Circuits,” *IEEE Journal of Solid-state Circuits*, vol. 21, no. 5, pp. 889–891, Oct. 1986.

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

[48] Shiva Shankar Ramani, and S. Bhanja, ”Any-time Probabilistic Switching Model using Bayesian Networks”, *International Symposium on Low Power Electronics and Design*, pp. 86-89, Newport Beach, CA, 2004.

[49] Z. Chen and K. Roy, “An Efficient Statistical Method to Estimate Average Power in Sequential Circuits Considering Input Sensitivities”, *ASIC Conference and Exhibit*, pp. 189–193, 1997.

[50] J. Cheng, “Efficient Stochastic Sampling Algorithms for Bayesian Networks,” *Ph.D Dissertation, University of Pittsburgh*, 2001.

[51] C. Yuan and M. J. Druzdzel, “An Importance Sampling Algorithm Based on Evidence Prepropagation,” *Proceedings of the 19th Annual Conference on Uncertainty on Artificial Intelligence*, pp. 624–631, 2003.

[52] M. Henrion, ”Propagating Uncertainity in Bayesian Networks by Probabilistic Logic Sampling”, In *Uncertainity in Artificial Intelligence 2*, pp. 149-163, New York, N.Y., 1988. Elsevier Science Publishing Company, Inc.

[53] P. Dagum and M. Luby, ”Approximating probabilistic inference in Bayesian belief networks is NP-hard”, *Artificial Intelligence*, 60(1):141-153, 1993.

[54] R. Fung and K.-C. Chang, ”Weighting and integrating evidence for stochastic simulation in Bayesian networks”, In M. Henrion, R.Shachter, L. Kanal, and J. Lemmer, editors, *Uncertainity in Artificial Intelligence 5*, pages 209-219, New York, N. Y., 1989. Elsevier Science Publishing Company, Inc.

[55] R. D. Shachter and M. A. Poet, ”Simulation approaches to general probabilistic inference on belief networks”, In M. Henrion, R.Shachter, L. Kanal, and J. Lemmer, editors, *Uncertainity in Artificial Intelligence 5*, pages 221-231, New York, N. Y., 1989. Elsevier Science Publishing Company, Inc.

[56] R. Fung and B. del Favero, ”Backward simulation in Bayesian networks”, In *Proceedings of the Tenth Annual Conference on Uncertainity in Artificial Intelligence (UAI-94)*, pages 227-234, San Mateo, CA, 1994. Morgan Kaufmann Publishers, Inc.

[57] K. Murphy, Y. Weiss, and M. Jordan, ”Loopy Belief Propagation for Approximate Inference: An Empirical study”, *Proceedings of the Conference on Uncertainity in AI(UAI-99)*, pp. 467-475, 1999.