A Complete Probabilistic Framework for Learning Input Models for Power and Crosstalk Estimation in VLSI Circuits

by

Nirmal Munuswamy Ramalingam

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.
Yun-Leei Chiou, Ph.D.
Srinivas Katkoori, Ph.D.

Date of Approval:
October 6, 2004

Keywords: Power Estimation, Learning Bayesian Network, Sampling, Crosstalk

DEDICATION

To my dad, mom, sister and my beautiful niece Shruti.

ACKNOWLEDGEMENTS

I like to first thank my advisor Dr. Sanjukta Bhanja, whose guiding light I followed throughout the project. She helped me in various ways as I hit the wall many a times through the course of this work. I would also like to thank Dr. Srinivas Katkoori and Dr. Yun-Leei Chiou for serving in my committee. My colleagues, the members of the VLSI Design Automation and Test Lab, namely Thara, Karthik (Bheem), Shiva, Sathish (Ponraj), Vivek (awk), also helped me with many insightful ideas, not to forget their company which I enjoyed while working in the lab.

Also, I would like to thank Bodka, Venky, Sathish for helping me with coding problems. I would also like to thank Melodie and Henry for making my work experince at the Division of Research Compliance a pleasant one. But it was the unconditional love from my parents and my sister that kept me going.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

A COMPLETE PROBABILISTIC FRAMEWORK FOR LEARNING INPUT MODELS FOR POWER AND CROSSTALK ESTIMATION IN VLSI CIRCUITS

Nirmal Munuswamy Ramalingam

ABSTRACT

Power disspiation is a growing concern in VLSI circuits. In this work we model the data dependence of power dissipation by learning an input model which we use for estimation of both switching activity and crosstalk for every node in the circuit. We use Bayesian networks to effectively model the spatio-temporal dependence in the inputs and we use the probabilistic graphical model to learn the structure of the dependency in the inputs. The learned structure is representative of the input model. Since we learn a causal model, we can use a larger number of independencies which guarantees a minimal structure. The Bayesian network is converted into a moral graph, which is then triangulated. The junction tree is formed with its nodes representing the cliques. Then we use logic sampling on the junction tree and the sample required is really low. Experimental results with ISCAS ’85 benchmark circuits show that we have acheived a very high compaction ratio with average error less than 2%. As HSPICE was used the results are the most accurate in terms of delay consideration. The results can further be used to predict the crosstalk between two neighboring nodes. This prediction helps in designing the circuit to avoid these problems.

CHAPTER 1

INTRODUCTION

Moore's Law which states that the number of transistors per integrated circuit would double every couple of years has been one of the driving forces for the development teams to break down these barriers. But the power consumed has not reduced and is not predicted to reduce even in the nano-domain. Power density or power dissipated per unit area is increasing due to the packing of millions of transistors in a single wafer (shown in Fig. 1.1.). Now, because of the increase in power density, cost increased for higher cooling and increased battery weight, not to forget the reduction in system reliability. Designers had to make low power devices to overcome these problems,and the impetus on low-power design, saw an increased attention to power estimation. The total power consumed by a circuit is called the average power is given by the formula,

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

CMOS circuits consist of a pull-up and pull-down network, which have a finite input fall/rise time larger than zero. During this short time interval, when both the pull-up and pull-down network are conducting, a current i_{cc} flows from supply to ground, called the short-circuit current, resulting in short circuit power.

Static leakage power dissipiation can be attributed to reverse bias diode leakage, sub-threshold leakage, gate oxide tunneling, leakage due to hot carrier injection, Gate-Induced Drain Leakage (GIDL), and channel punchthrough. Note that this type of power dissipation depends on the logic states of a circuit than its switching activites.

The most important is the dynamic component which constitutes about 80% of the average power. The current i_d that flows due to the charging and discharging of the parisitic capacitance during switching causes the power dissipation $P_{dynamic}$.

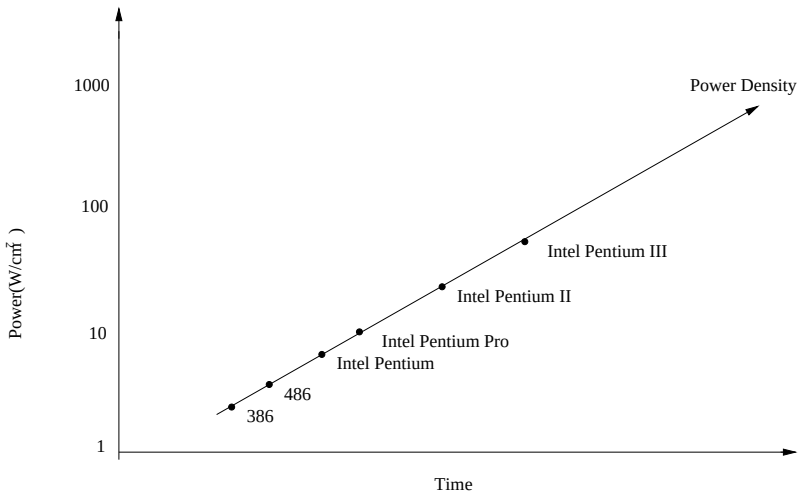

Figure 1.1. Power density as predicted in [26].

Having a gate level implementation of a target circuit, to estimate the total power dissipation, we can sum over all the gates in the circuit the average power dissipation due to capacitive switching currents, that is:

$$P_{avg} = 0.5 f_{clk} V_{dd}^2 \sum_n (C_n.sw_n) \tag{1.2}$$

where f_{clk} is the clock frequency, V_{dd} is the supply voltage, C_n and sw_n are the capacitance and the average switching activity of the gate n, respectively. From this equation we can see that the average switching activity of every gate in the circuit is a key parameter that needs to estimated accurately, particulalry if one needs the node-by-node power as the voltage and clock frequency are known to the designers. Switching activity is a measure for the number of gates and their outputs that change their bit-value during a clock cycle. The toggling between logic zero and logic one, capacitances get charged and discharged.

1.1 Power Estimation

Average power estimation techniques can be divided into three broad categories, namely estimation by simulation, statistical simulation and probabilistic techniques as shown in Fig. 1.2.

Simulation based power analysis requires a set of simulation vectors at the primary inputs of the circuit to trigger the circuit activities. To obtain the power dissipation of the circuit, the switching activity information is collected and applied to power models after simulation. These vectors have

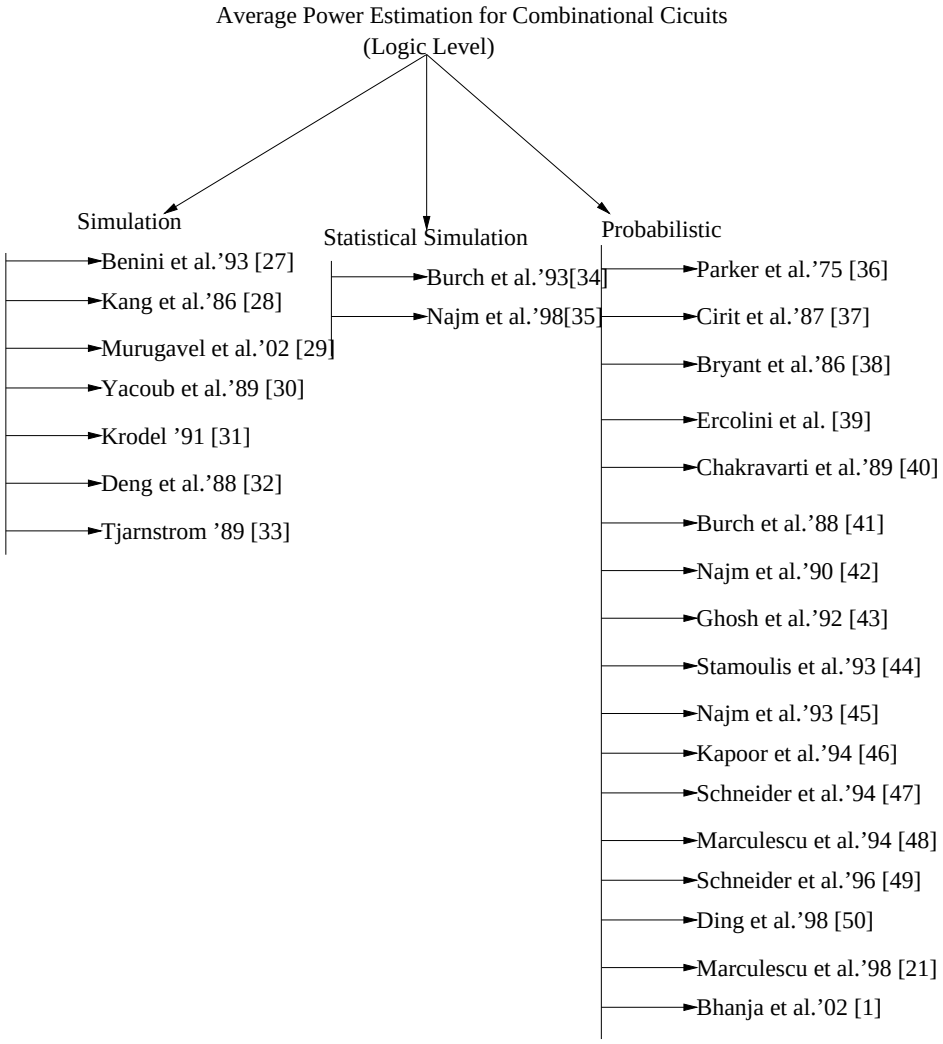

Figure 1.2. Power Estimation Techniques.

substantial impact on the power values because the power dissipation relies heavily on the switching activity. Each vector causes some energy to be dissipated and the total power is the summation of the energy of each vector and dividing over the total simulation time. Usually these techniques provide sufficient accuracy at the expense of large running times as these methods use a very large set of input vectors. However the method becomes unrealistic to rely on when done on large circuits.

Statistical methods are used in combination with simulation techinques in statistical simulation [41], [42] to determine the stopping criterion. Though these techniques are efficient in terms of the time required, one has to be careful in modeling statistical patterns at the inputs and care should be taken to not get trapped in a local minima.

In probabilistic techniques the input statistics (e.g., switching activity of the inputs, signal correlations, etc.) are first gathered interms of probabilities and then they are propagated. They are fast and more adaptable, but involve an assumption about joint correlations. They provide sufficient accuracy with a low computational overhead, but factors such as slew rates, glitch generation and propagation are difficult to capture. Another challenge in the method is the ability to account for internal dependencies due to reconvergent fan out of the target circuit.

1.2 Input Model

It is crucial that the vectors used in simulation represent the typical conditions at which the power estimate is sought. If the simulation vectors do not contain the proper instruction mix the power analysis result will be skewed. Regardless of how the simulation vectors are generated, if we simulate the circuit with only several vectors, the power dissipation result obtained is not truthful because the vector length is too short. Most part of the circuit is probably not exercised enough to obtain actual toggling activites. On the other hand we can simulate the circuit for a very large number of vectors to obtain an accurate measure of the power dissipation. But is it necessary wasting computer resources to simulate that many vectors? How much extra accuracy can we acheive by simulating a million vectors versus only a thousand vectors? How do we know that we have simulated enough vector length? What is the cost involved?

As said in [5] the vector compaction problem reduces the gap between simulative and nonsimulative approaches. The input statistics that must be captured and the length of the input sequences which must

be applied are some of the issues that must be taken into consideration. Generating a minimal length sequence that staisfies the above conditions is not a trivial task. So vector compaction is the technique by which a smaller set of vectors is derived from a larger set by a series of steps such that the smaller derived set preserves the original statistical properties of the larger initial set.

Coming back to the question on the length of the vectors, simulation methods are accurate but suffer cost and memory overhead, which limit the size of the input vector set to hundreds or thousands of vectors. But this results in inaccuracy in the power estimation process because the power consumption in digital circuits is input pattern dependant, that is depending on the input vectors applied to the target circuit very different power estimates may be obtained. To obtain an accurate power estimate, a set of inputs that resemble the characteristics of the data for typical applications is required. So to acheive this goal the vector set has a size of millions of vectors. Vectors of the size of hundreds or thousands, if selected randomly from the large vector set may not be able to capture the typical behavior and thus may lead to an underestimation or overestimation of the power consumed by the circuit. So our method of vector compaction solves the problem by compacting the millions of vectors into a characteristic input set of a much smaller length, yet statistically equivalent to the original larger vector, thus provding a power estimate very close to the power estimated by the larger sequence.

Switching model of a VLSI circuit is a comprehensive representaion of switching behavior of all the signals in the circuit. At each signal we store the state at time t and the state at time $t-1$. The dependency among all the switching variable can be captured as a joint Probability distribution function. One might look into switching model as a way to establish the role of data as inputs to the circuits. In essence, the switching model captures the data-driven uncertainty in VLSI circuits in a comprehensive probabilistic framework. Needless to say that, modeling inputs become an integral part of the switching model, even though a handful of prior work in power or switching analysis, has modeled inputs efficiently. Even analysis that are vector driven (simulation and statistical simulation) have mostly become assumed input priors as random inputs. In this work, we take a look into modeling inputs through a causal graphical probabilistic approach which models input space in a compact way. This input model then can be fused with a vectorless probabilistic model or be used to generate samples for statistical or simulative approaches by a random walk in the probabilsitic network.

Note that the switching model is extremely relevant of both static and dynamic component of power as shown in Eq. 1.3. In Eq. 1.3, P_{dg} represents the dynamic component of power at the output of a gate g. The impact of data on dynamic component of power is encasulated in α, the singleton switching activity. The static component of power P_{sg} is dominated by $P_{leak,i}$, leakage loss in a leakage mode i. It has to be noted that each leakage mode is determined by the steady state signals that each transistor in the gate would be in. For example, in a two input(say A and B) NAND gate, the gate would have four dominant leakage mode (i=4: $A_{@0}B_{@0}, A_{@0}B_{@1}, A_{@1}B_{@0}$ *and* $A_{@1}B_{@1}$). β is the probability of each mode i. Probabilistically both α and β are singleton probability of switching and joint probability of multiple signals in a gate respectively and are dependent on the input data profile. The switching model is affected by various factors such as the topology of the circuit, the input statistics, the correlation between nodes, the gate type, and the gate delays, thus making the estimation process a complex procedure.

$$\begin{aligned} P_t &= \sum_g P_{tg} = P_{dg} + P_{sg} \\ &= 0.5\alpha f V_{dd}^2 C_{load+wire} + \sum_i P_{leak,i}\beta_i \end{aligned} \tag{1.3}$$

In this work, we focus on modeling the inputs and generate a probabilistic structure among the inputs that can be used to study the behaviors of internal nodes. Even though estimation of singleton switching has been discussed in great depth and many of the procedures are input driven (simulative and statistical simulative), inputs are studied in a limited set of works [2, 3, 57].

Eq. 1.4 demonstrates the effect of input model. Let X_i be an intermediate signal which could be singleton switching variable representing switching states ($0_{t-1}0_t, 0_{t-1}1_t, 1_{t-1}0_t$ *and* $1_{t-1}1_t$), of a signal or X_i could be composite signals (A,B) in a transistor stack and can have composite states namely $A = 0_{t-1}0_t,\ B = 0_{t-1}0_t$. In estimating X_i, as shown in Eq. 1.4, we have two components, namely the set $P(X_i|\{I_j\})$, $\{I_j\}$ denoting the primary inputs of the circuits, where j is the jth input and $P(\{I_j\})$ a prior probability of the the input space. The first component can be analysed by probabilistic measure considering a joint pdf of the entire signals through a probabilistic framework as in [43, 21, 1] or can be measured '[28, 20] for specific input patterns. The second component $P(\{I_j\})$ however, is important for both stimulus-sensitive approach and stimulus-free approach where we need to model the dependency structure of the inputs and then either use it in the joint pdf of the signals for probabilistic methods or

use it to generate representative specific input samples for measurement-based estimates. The theme of this work is to obtain correct priors in the input space $P(\{I_j\})$ and then to generate samples that closely emulate the behavior of the input space or to fuse the prior on to an existing probabilistic set-up. Note that, we do not concentrate on compaction, but our claim is that the samples are so close to the actual distribution that the sample requirements are really low and a high compaction ratio is automatically achieved.

$$P(X_i) = P(X_i|\{I_j\}).P(\{I_j\}) \tag{1.4}$$

Note that input modeling is not a simple task and can be useful in other areas like test vector and pattern generation. This work is an effort to learn a causal structure in the input data. Causal structures are common to obtain for real-life data and have been proven successful in modeling complex data-sets like gene-matching and speech processing. A causal model can encapsulate two additional independencies (induced dependency among the cause of a common child, and weakly transitive dependency) over and above their undirected Markov models that makes it specially attractive in reducing the structure. To our knowledge, this is the first model where causality in the data is utilized in learning a structure for an input model in VLSI.

We use dependency analysis in inputs to arrive at a Bayesian Network which involves three steps, drafting, thickening and thinning as done in [16]. In the first step mutual information between pairs of nodes is calculated and an approximate graph based on the information is created. In the next step arcs are added when the pairs of nodes are not conditionally independent on a ceratin condition set. If the model is DAG faithful, the graph obtained here is an I-map of the model. In the last step each arc added is checked using conditional independence tests and will be removed if the nodes are conditionally independent. The final result is a P-map of the underlying model.

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 but also serves as a computational mechanism for efficient probabilistic walk generating samples.

Probabilistic Logic Sampling is done inside the cliques of the junction tree to obtain the necessary samples. The steps involved in formation the juction tree is the creation of a moral graph, and the process is called compilation, then the moral graph is triangulated. The clique set is identified and the junction tree of cliques is formed. Given a Bayesian network, a moral graph is obtained by 'marrying parents', that is adding undirected edges between the parents of a common child node. Before this step, all the directions in the DAG are removed. The moral graph is said to be triangulated if it is chordal. The undirected graph G is called chordal or triangulated if every on of its cycles of length greater than or equal to 4 posesses a chord [9], that is we add additional links to the moral graph, so that cycles longer than 3 nodes are broken into cycles of three nodes. The juntion tree is defined as a tree with nodes representing cliques (collection of completely connected nodes) and between two cliques in the tree T there is a unique path. Probabilistic Logic Sampling is a method proposed by Henrion [56] which employs a stochastic simulation approach to make probabilistic inferences in large multiply connected networks. If we represent a Bayesian network by a sample of m deterministic scenarios s=1, 2,.....m and $L_s(x)$ is the truth of event x in scenario s, then uncertainty about x can be represented by a logic sample.

1.3 Crosstalk Estimation

An evaluation node is a circuit node that forms a connection between channel connected components in the design. Noise can be defined as anything that causes the voltage of an evaluation node to deviate from the nominal supply or ground rails when it should normally have a stable high or low as determined by the logic of the circuit in consideration. The noise sources that are of interest to digital design are leakage noise, charge-sharing noise, power supply noise and crosstalk noise.

Leakage is due to the off current of FETs, and is largely due to the subthreshold current and is directly determined by the threshold voltage and temperature. Whereas power supply noise is the noise appearing on the supply and ground nets and coupled onto evaluation nodes through a FET conduction path. Charge sharing noise is caused due to the charge redistribution between the dynamic evaluation node and internal nodes of the circuit. Crosstalk noise is the voltage induced on a node

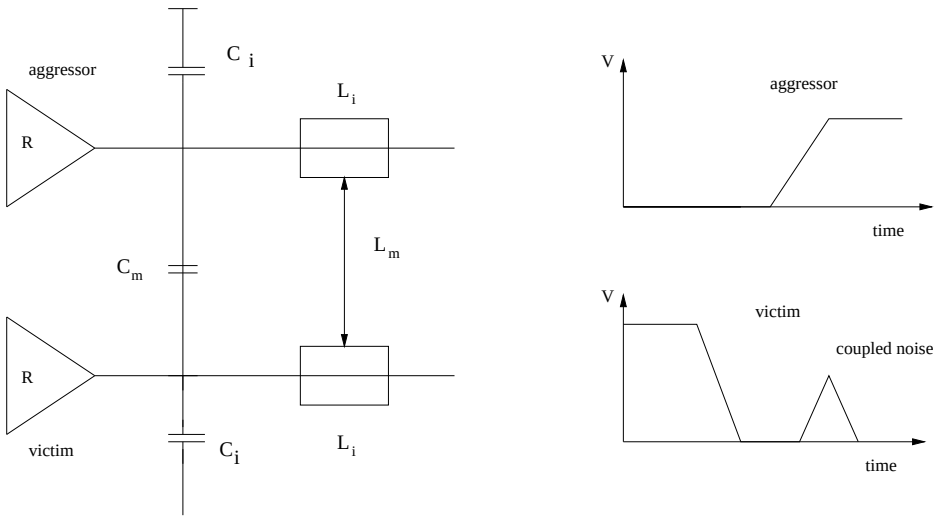

Figure 1.3. Simple Model of Coupled Wires.

due to capacitive coupling to a switching node of another net. It can also be said as the capacitive and inductive interference caused by the node voltage developed on signal lines when the nearby lines change state. It is a function of the separation between signal lines, the linear distance that signal lines run parallel with each other. In today's logic devices the faster edge rate greatly increases the possibility of coupling or crosstalk between the signals. Crosstalk is one of the issues that hamper designers to acheive higher speeds, and so it must be reduced to levels where no extra time is required for the signal to stabilize. The importance to estimate crosstalk can be further understood as we discuss the faults that it may be induced between two coupled interconnects, namely the aggressor and the victim.

1. *Delay Fault*

   It occurs when signals of the two coupled interconnections undergo opposite swings. This fault affects the gate delay which inturn can change the critical path delay and cause glitches.

2. *Logic Fault*

   Here a logic error occurs when the voltage induced in the victim interconnect by the aggressor interconnect is greater than a threshold. This causes the circuit malfunction when the risk tolerance bound is exceeded.

3. *Noise-induced Race Failures*

The race failures are a consequence of the delay fault. The change in the delay causes a hold-time violation, commonly noticed in pipelined circuits.

In a simplified model as shown in Fig. 1.3., coupling can be considered between the two lines as a capacitive voltage divider. Higher integration causes larger mutual capacitance C_m and smaller intrinsic capacitance C_i of a line, both worsening cross talk. The same is valid for the inductive coupling as the mutual inductance L_m grows significantly in denser structures. Taking the output resistances R and the input capacitances of the receiving circuits into account the model shows a crosstalk voltage propotional to the coupling wire length.

1.4 Contribution of the Thesis

The contribution of this work is two-fold. First, we arrive at a probabilistic graphical model in the inputs that is (i) edge-minimal (ii) exact in terms of dependence and (iii) easy to learn. Second, it is elegant as a model and also as a source of generating samples from this graphical probabilistic structure that closely resemble the dependency in the inputs and a few samples converges to the mean of the underlying distribution. Thus, this input-model can be of dual purpose: (i) It can be fused with a graph-based probabilistic set up for the circuit dependency [1] and make the whole estimation process stimulus-free and (ii) can be used to generate samples that closely match the original dependency model in inputs for statistical simulation and simulation based estimation.

The salient features of the proposed Bayesian Network (BN) learning model for inputs are as follows.

1. *It generates an edge-minimal structure that models dependency exactly under a causal data environment*
2. *The computations are easy and learning algorithms are O(N^2) to O(N^4) in terms of number of inputs*
3. *The dependency model of the inputs can be fused with grphical structure of the internal circuit making the estimation stimulus-free and insensitive to measurements*

4. *The dependency model can be probabilistically efficiently sampled such that samples closely emulate the dependencies in the inputs for statistical simulation and simulation based estimation process.*
5. *The performance that is seen the ISCAS circuits generates a maximum error of 1.8% with a compression ratio up to 300.*

CHAPTER 2

RELATED WORK

2.1 Vector Compaction

The pioneering work in this field done by Marculescu et. al, [2]-[7], have approached the problem first using a Stochastic Sequencial Machine and later by using Markov Networks.

In [3] the approach to tackle this problem was based on the stochastic sequential machines(SSMs) theory and emphasis was on those aspects related to Moore-type machines. Finite Automata are mathematical models for various systems with a finite number of states. These models accept particular inputs, at discrete time intervals and emit outputs accordingly. The automata is considered to show stochatic behavior as the current state of the machine and the input given determine accordingly the next state and the output of the automaton. Here the authors synthesize first a stochastic machine that is probabilistically equivalent to the initial sequence. Then, by applying randomly generated inputs to the stochastic machine, a random walk through the states of the transition graph is done and a shorter sequnce is generated. The idea proposed in this paper is that the automaton when in state x_i and receiving input n, can move into a new state x_j with a positive probability $p(x_i, n)$. The basic task is of synthesizing an SSM which is capable of generating constrained input sequences and use it in power estimation. The method proposed for power estimation of a target circuit for a given input vector sequence of length L_0 is to derive a probabilistic model based on SSMs and generate a shorter sequence L, which is used to calculate the power of the target circuit.

The method used in [3] has some limitations as the authors use probabilistic automata theory to synthesise stochastic machines, where the compaction technique becomes a multi-step process. An initial pass through the sequence is performed first to extract the statistics needed and then the stochastic machine is synthesised to generate the new compacted sequence. This method bears a huge load on computer memory and time especially when vector sequences are very large. Also, some distributions

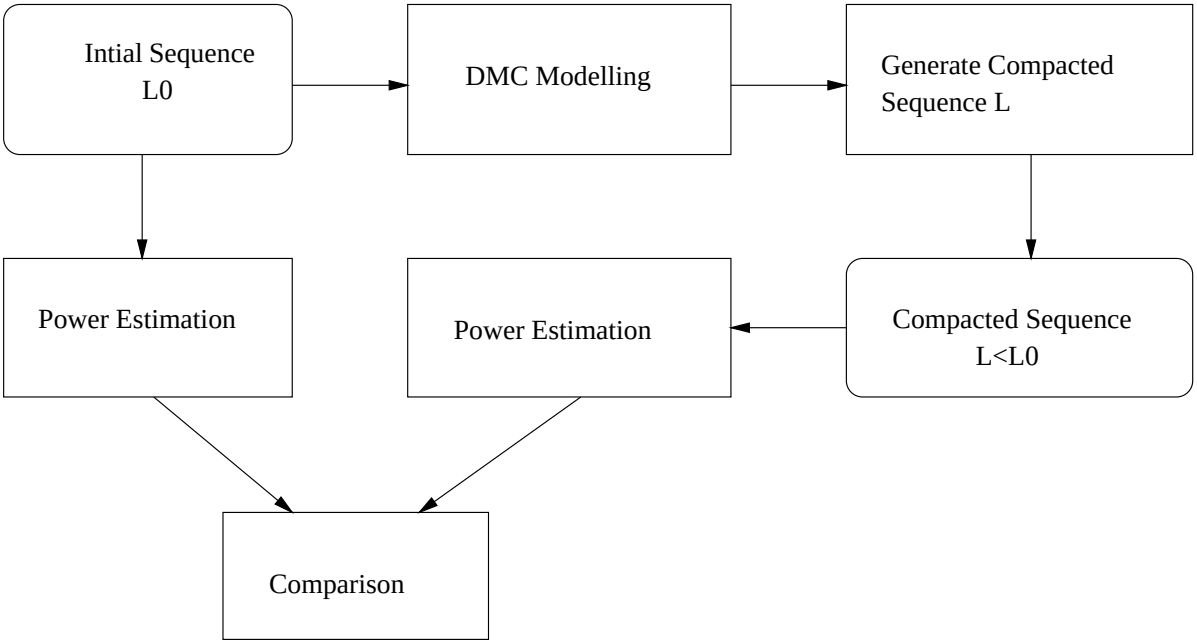

Figure 2.1. DMC Modeling.

of the transition probabilities tend to lean towards a small number of degenerate matrices, in contrast to the others which take much longer decompositions. Since the values in the initial matrix themselves are important in the decomposition process, the decomposition part of the matrix becomes an important step as far as the running time is concerned. Here, therefore one should allow limited precision in the calculations to simplify the decomposition process. These disadvantages are taken into consideration in [4]-[7], with the use of DMC modeling. Here to estimate the power consumption of a target circuit for a given input sequence L1(Fig. 2.2.a), a Markov model of the input sequence is first derived through one-pass traversal technique and then, with the help of this compact representation a smaller sequence L, equivalent to L1 is generated, which is used in power simulators to determine the power consumption of the circuit,(Fig. 2.2.b). After having the initial sequence lossy compression method is used to generate the compacted sequence. The foundation of the approach here is dependent on the adaptive modeling of the binary input streams as first-order Markov sources of information.

In [6], another method called the hierarchical Markov modeling is introduced. Though the foundation of the approach is probabilistic and relies on adaptive modeling of binary input streams as first-order Markov sources of information, hierarchical Markov models are used to structure the input space into a hierarchy of micro and macro states. The first(high) level in the hierarchy are the Markov

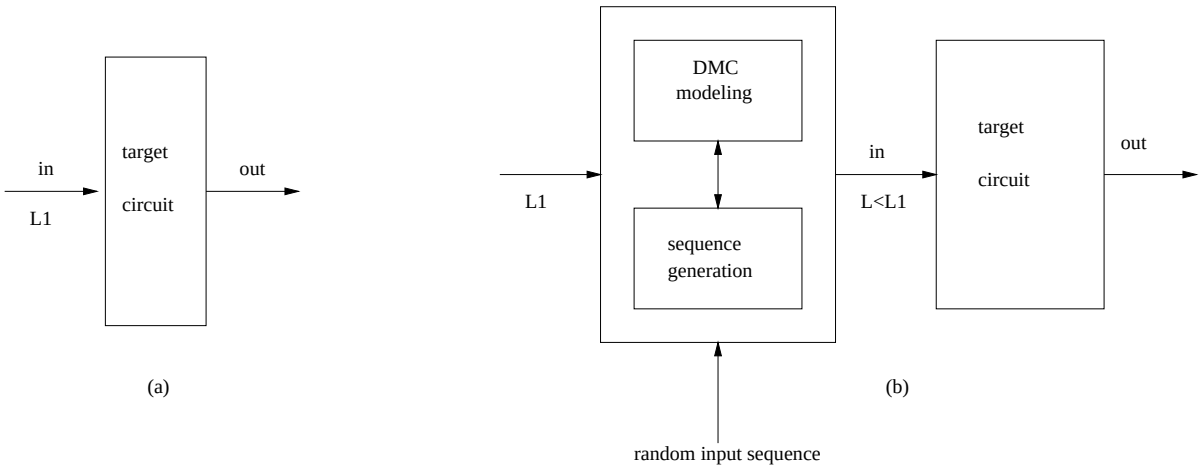

Figure 2.2. Data Compaction for Power Estimation.

chain of macrostates and at the the second(low) level, each macrostate is in turn characterized by a Markov chain for all its constituent microstates. After the hierarchy for an input sequence is constructed, starting with some macrostate, a compaction procedure with the needed compaction ratio is applied to compact the set of microstates within that macrostate. Then the control returns to a higher lever hierarchy and based on the conditional probabilities that characterize the Markov chain at this level, an new macrostate is entered and the process is repeated. In [2] two different techniques are proposed to accomodate temporal compatability. The first method discards the temporal incompatibility between the pairs of consecutive vectors. The next method adresses this problem by making sure the consecutively generated vectors are temporally compatible, by proposing a greedy mechanism.

The compaction technique used in [8] is based on fractal concepts. The corrrelation in the input vectors is exploited and the algorithm divides the larger vector set into smaller fractal subsets. Then the fractal subsets that are similiar to a particular subset are removed from the original vector set, and thus obtaining a compacted vector set. The authors in [12] catergorize the input pattern pairs into several groups according to their power characteristics. Then the shorter sequence is obtained by consecutive sampling. In the method proposed in [14], a set of test vectors is generated using any test generation algorithm. Then the compaction procedure, COMPACT is used, and the tests obtained here, are compared to the other tests which are generated previously. If the set generated by COMPACT is compatable with the other tests then the latter set is replaced by the common vectors. Three dynamic test vector compaction methods are specified here in [15] for a reduced test sequence generation. In

all the methods the sequence generated from an Automatic Test Pattern Generator(ATPG) for a single fault, is assigned specific values for unspecified primary inputs. In the first method called the Best Random Fill, all the unspecified primary inputs are given finite number of random numbers, and each one simulated to determine which detects the most number of faults. The next method called the Best Random Fill 2 has slight modifications of the previous algorithm. Here each fault is given a weight, the harder one getting the highest, and the fill which gets the highest number of harder-to-find faults is selected. This algorithm is made more effective by not producing similar random fills. The next method known as the Weighted Deterministic Fill, gives deterministic values to the unspecified inputs. First the number of stuck-at-0 and stuck-at-1 faults propogated is found. Then the gate with highest number of faults passed to it is found and its inputs are assigned by backtracking to primary inputs.

2.2 Crosstalk Estimation

In [24] the authors aim to find the set of interconnection pairs which will not pose crosstalk problems during normal operation of the circuit. They state that the effect of the environment upon any target circuit, more specifically on its layout that it is designed to work, and under conditons which are assumed to be known to the designer can be modeled by capturing the input pattern dependencies. Taking into account this type of modeling can result in a much simpler circuit than when it designed discarding the input pattern correlations. In [25] the necessity to consider on-chip simultaneous switching noise (SSN) is dealt with, particularly when determining the propagation delay of a CMOS logic gate in a high speed synchronous CMOS ICs. The paper presents analytical expressions which characterize on-chip SSN and also the effect of SSN on the propagation delay and output voltage of a logic gate. In [51] the authors extract the electrical parameters for a base set of adjacent wire configuration. This step is done only once dependent on the the technology, then the geometry of the complete chip routing spaces is extracted. The adjacency lengths between the chosen victim and its agressors considering the wire widths and spacings is calculated. Then the effective ouput resistances influencing the signal slopes and the amount of coupled noise is determined. Finally the switching times of victim and agressor lines from the chip's timing report is read and all timing-uncritical wire adjacencies are disregarded. A comprehensive methodolgy for understanding and analyzing the noise immunity of digital integrated circuits is presented in [52]. A noise classification based on noise level relative to the

supply and ground rails is introduced. A noise stability metric as a practical formal basis for ensuring noise immunity is described. A static noise anlaysis approach is also explained, which can be used as a technique for identifying all possible on-chip functional failures without full pattern dependent dynamical simulation. The Harmony is then used to combine static timing analysis and reduced order modelling with transistor-level analysis.

CHAPTER 3

LEARNING BAYESIAN NETWORKS

3.1 Bayesian Networks

Bayesian Network is a graphical probabilistic model based on the minimal graphical representation of the underlying joint probability function. It can be defined as a directed acyclic graph(DAG) with a probability table for each node and the nodes in the network represent the propositional(random) variables in the domain and the arcs between the nodes represent the dependency relationship among the variables. Before delving deep into the definition let us discuss some basic concepts.

Definition: Let E and F be events such that $P(F) \neq 0$. Then the conditional probability of E given F, is given by

$$P(E \mid F) = \frac{P(E \cap F)}{P(F)} \tag{3.1}$$

Bayes' Theorem : The Bayes theorem devoleped by Thomas Bayes in 1763 forms the basis of Bayesian Networks.

Definition: Given two events E anf F such that $P(E) \neq 0$ and $P(F) \neq 0$, then

$$P(E \mid F) = \frac{P(E \mid F)P(E)}{P(F)} \tag{3.2}$$

Proof: we use Eq. 3.1 to derive the Bayes theorem.

$P(E \mid F) = \frac{P(E \cap F)}{P(F)}$ and similarly we can write,

$P(F \mid E) = \frac{P(F \cap E)}{P(E)}$.

Multiplying these equalities by the denominator, we get,

$P(E \mid F)P(F) = P(F \mid E)P(E)$ as $P(E \cap F) = P(F \cap E)$, so

$P(E \mid F) = \frac{P(E|F)P(E)}{P(F)}$.

Some basic definitions related to Bayesian networks are as given next:

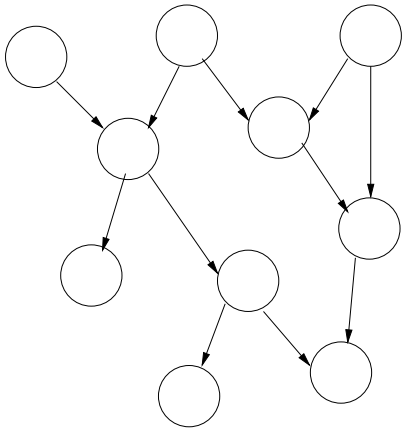

Figure 3.1. Example for a DAG.

Definition 1 : A *directed graph* G denoted as (V,E), where V is a finite, non-empty set whose elements are called vertices or nodes, and E is a set of ordered pairs of distinct elements of V. Elements of E are also called the edges or arcs. If (x,y)$\in$E, we can say that there is a directed edge from x to y and that x and y are incident to the edge. It is denoted by an arrow from x to y , and we can say that x and y are incident to the edge.

x and y are said to be adjacent or neighbors if there is an edge from x to y or from y to x. If the start of the arrow is at x and the end point at y, then x is called the *parent* of y and y is called the *child* of x. Similarly x is called the ancestor of y and y the descendent of x. The set of edges connecting the nodes x and y is called the *path* from x to y.

A *directed cycle* is a path from a node to itself. A *simple path* is one with no subpaths which are directed cycles.

Definition 2: A directed graph that contains no directed loops or cycles is called a *directed acyclic graph(DAG)*.

An example of a Directed Acyclic Graph(DAG) is shown in Fig. 3.1.

Definition 3: Let U be a finite set of discrete value variables. Let $P(.)$ be a joint probability function over the variables in U, and let X,Y,Z be any three subsets of variables in U. X and Y are said to be *conditionally independent* given Z if $P(x \mid y,z) = P(x \mid z), if P(y,z) > 0$. Conditional independence means that knowledge of Z makes X and Y independent of each other.

Definition 4: A DAG G is a *dependency map (D-map)* of a dependency model M, if every independence relationship derived from M can be expressed in G.

Definition 5: A DAG G is an *independence map (I-map)* of a dependency model M, if every independence relationship derived from G corresponds to a valid conditional independence relationship in M.

Definition 6: A DAG G is a *minimal I-map* of a dependency model M if it is an I-map of M and deletion of any of its edges from G destroys the independence relation of M.

Definition 7: A DAG G is both a D-map and an I-map of the dependency model M, if it is a *perfect map (P-map)* of M.

Definition 8: A DAG G is called a *Bayesian Network* of a probability function P on a set of variables U, if G is a minimal I-map of P.

Also a Bayesian network is a directed acyclic graph (DAG) representation of the conditional factoring of a joint probability distribution. Any probability function $P(x_1, \cdots, x_n)$ can be written as[1]

$$P(x_1, \cdots, x_N) = P(x_n|x_{n-1}, x_{n-2}, \cdots, x_1)P(x_{n-1}|x_{n-2}, x_{n-3}, \cdots, x_1) \cdots P(x_1) \tag{3.3}$$

This expression holds for any ordering of the random variables. The representation and infernce using this equation becomes very tedious when n becomes very large. In most applications, a variable is usually not dependent on all other variables. There are lots of conditional independencies embedded among the random variables. The above equation does not consider these independecies, but these dependencies can be used to reorder the random variables and to simplify the conditional probabilities.

$$P(x_1, \cdots, x_N) = \Pi_v P(x_v|Pa(X_v)) \tag{3.4}$$

where $Pa(X_v)$ are the parents of the variable x_v, representing its direct causes. This factoring of the joint probability function can be represented as a directed acyclic graph (DAG), with nodes (V) representing the random variables and directed links (E) from the parents to the children, denoting direct dependencies.

Definition 9: A Bayesian network G of a distribution P determines a set of independence relations. These independence relations in turn may entail others, in the sense that every probability distribution having independence relations will also have further independence relations as explained in [18]. In

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

other words G of the distribution P may not represent all the conditional independence relations of P. But, when G can actually represent all the conditional independence relations of P, then P and G are faithful to each other. In [10], G is called a perfect map of P and P is called the *DAG-Isomorph* of G.

Definition : A dependency model M is said to be causal or a DAG isomorph if there is a DAG D that is a perfect map of M relative to d-separation, i.e.,

$$I(X,Z,Y)_M \Longleftrightarrow < X \mid Z \mid Y >_D \tag{3.5}$$

Theorem : A necessary condition for a depndency model M to be a DAG isomorph is that $I(X,Z,Y)_M$ satisfies the following independent axioms.

Symmetry :

$$I(X,Z,Y) \Longleftrightarrow I(Y,Z,X) \tag{3.6}$$

Composition/Decomposition :

$$I(X,Z,Y \cup W) \Longleftrightarrow I(X,Z,Y) \& I(X,Z,W) \tag{3.7}$$

Intersection :

$$I(X,Z \cup W,Y) \& I(X,Z \cup Y,W) \Longrightarrow I(X,Z,Y \cup W) \tag{3.8}$$

Weak union :

$$I(X,Z,Y \cup W) \Longrightarrow I(X,Z \cup W,Y) \tag{3.9}$$

Contraction :

$$I(X,Z \cup Y,W) \& I(X,Z,Y) \Longrightarrow I(X,Z,Y \cup W) \tag{3.10}$$

Weak Transitivity :

$$I(X,Z,Y) \& I(X,Z \cup \gamma,Y) \Longrightarrow I(X,Z,\gamma) or I(\gamma,Z,Y) \tag{3.11}$$

Chordality :

$$I(\alpha,\gamma \cup \delta,\beta) \& I(\gamma,\alpha \cup \beta,\delta) \Longrightarrow I(\alpha,\gamma,\beta) or I(\alpha,\delta,\beta) \tag{3.12}$$

Why Bayesian Networks?

1. Bayes Networks provide a natural representation for conditional independence.
2. Topology and the Conditional probability tables provide a compact representation of the joint distribution.
3. Bayesian networks are generally easy to construct. An equivalent representation namely Markov networks are tedious to construct and moreover they are undirectional and inferencing becomes more time and memory consuming.
4. Inference in Bayes networks is easier, for example polytree inferencing is an NP hard problem on general graphs.
5. Learning of Bayesian networks provide a compact representation of the data. Having this representation of the data, inferencing is easier and faster.

3.2 Learning

In a Bayesian network, the graph, called the Directed Acyclic Graph(DAG), is the structure of the Bayesian Network, and the conditional probability distribution is called the parameter. Both the parameter and the structure can be seperately learned from the data. While learning the parameter, we assume that we know the structure of the DAG, but in structure learning we start with only a set of random variables with unknown relative frequency distribution. Learning structure can also be done with missing data items, and hidden variables and also in the case of continuous variables as discussed in [11]. There are two major methods to Bayesian network learning, namely search and scoring method and dependency analysis methods. In this experiment we are using the dependency analysis method to learn the bayesian network structure from data. The input to learn the Bayesian network is a database table. Each node in the network is a representative of the fields in the database. Each record in the given database is a complete instantiation of the random variables. We assume that the database table has discrete values and the data set is complete with no missing values. Also, the volume of the data set should be large enough so that reliable CI tests could be performed. In dependency analysis method, conditional independence play an important role, and by using the concept of direction dependent

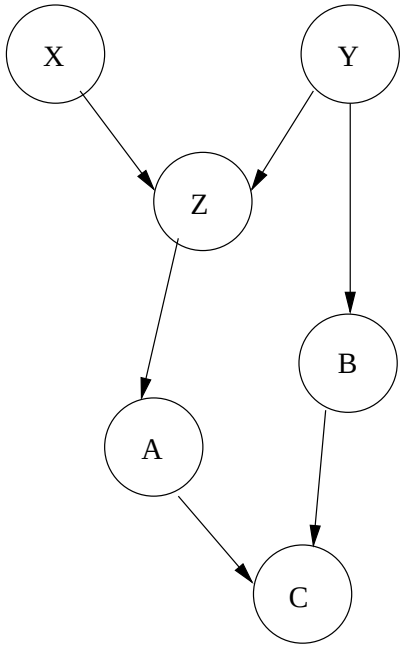

Figure 3.2. Example for d-separation.

separation or d-separation (Pearl, 1988), all the independence relations of the Bayesian network can be computed.

Definition: For a DAG $G = (V,E)$, where X,Y $\in V$ and $X \neq Y$, and $C \subset V \setminus X,Y$, we say that X and Y are d-seperated given C in G if and only if there exists no adjacency path P between X and Y, such that :

(i) every converging arrow on P is in C or has a descendent in C and

(ii) no other nodes on path P is in C. C is called the cut-set. If X and Y are not d-seperated given C, we say that X and Y are d-connected given C.

In more simple terms, we can explain d-separation with the an example as shown in Fig. 3.2..

A simple explanation is given as, two nodes are d-separated given the third if all active paths from the first node to the second node are blocked given the third. It can also be explained as, if knowledge about the first node gives no extra information about the second once the third node is known. That is, once we know about the third node, the first node adds nothing to what we already know about the second node.

In the exampe, Fig. 3.2. Z is d-separated from B, given Y as all paths from X to Y are blocked given Z. Also Z is not d-separated or it is d-connected from B given C as all paths between Z and B are not blocked as there exists a path through Y.

If two nodes are dependent, then the knowledge of the value of one node will give us some information of the value of the other node. This information gain can be measured by using mutual information. Therefore the knowledge of mutual information can tell us about the dependency relation between two nodes. The mutual information of two nodes X_i,X_j is expressed as:

$$I(X_i, X_j) = \sum_{x_i, x_j} P(x_i, x_j) \log \frac{P(x_i, x_j)}{P(x_i)P(x_j)} \tag{3.13}$$

and the conditional mutual information is defined as

$$I(X_i, X_j \mid C) = \sum_{x_i, x_j, c} P(x_i, x_j, c) \log \frac{P(x_i, x_j \mid c)}{P(x_i \mid c)P(x_j \mid c)} \tag{3.14}$$

where C is a set of nodes. When $I(X_i, X_j)$ is smaller than a certain threshold ε, we say that X_i, X_j are marginally independent. When $I(X_i, X_j \mid C)$ is smaller than ε, we say that X_i, X_j are conditionally independent given C.

3.3 Algorithm for Learning Bayesian Network Given Node Ordering

The algorithm is the work of **Cheng et.al** [16] which constructs a Bayesian network based on dependency analysis from a database table as input. A similar work is the Chow-Liu algorithm as explained in [10].

There are three steps, that the algorithm performs, namely drafting, thickening and thinning. In the first step a draft is created based on the mututal information of each pair of nodes. In the second step known as thickening, arcs are added when the pair of nodes are not conditionally independent based on a certain condition set. We get an I-map at the end of this step which is of the underlying dependency model given it is DAG faithful. In the last step the arcs added in the previous step are checked using conditional independence tests and the arc will be removed if any two nodes are condtionally independent. So at the end of the third step we get a perfect map of the model when it is DAG faithful.

3.3.1 Step 1:Drafting

We initiate a graph $G(V,E)$, where V has all the attributes of a data set and E is an empty set. We also inititate an empty list L. For $(v_i, v_j \in V$ and $i \neq j$, for each pair of nodes (v_i, v_j), we compute mutual information using equation 3.13. Then we sort the pairs of nodes in ascending order that have mutual information greater than a certain small value ε and are put into the list L. The pointer p points to the first pair of nodes in L. Then the first pair of nodes is removed from L and the arcs are added to E, and the direction of the arc depends on the node ordering. The pointer is moved to the next pair of nodes. We get the pair of nodes pointer by L, and if there is no open path , the corresponding arc is added to E, and the pair of nodes is removed from L. These steps are repeated till the we cover all pairs of nodes. The reason to sort the mutual information in ascending order is heuristic, which states that higher mutual information represents a direct connection than a lower one. It has been proven correct if the underlying graph is a singly connected graph. A flow chart representation of the steps is shown below.

This step tries to find a draft which is more like the final model as much as possible by only using pairwise mutual information tests without involving conditional independence tests. The result obtained at the end of the step can be anything from an empty graph to a complete graph without affecting the final graph at the end of all the steps of the algorithm. The draft learning procedure comes to a stop when every pairwise dependency is expressed by an open path in the draft.

The working mechanism cane be graphically illustrated by the following simple multi-connected network borrowed from [18].

For each pair of nodes we get the mutual information, as explained before. In the example, suppose we have $I(B,D) \geq I(C,E) \geq I(B,E) \geq I(A,B) \geq I(B,C) \geq I(C,D) \geq I(D,E) \geq I(A,D) \geq I(A,E) \geq I(A,C)$, and these pairs have mutual information greater than ε. Now the list L contains [{B,D}, {C,E}, {B,E}, {A,B}, {B,C}, {C,D}, {D,E}, {A,D}, {A,E}, {A,C}]. Next we get a pair of nodes iteratively from the list L, and connect the two nodes by an arc, and remove the pair from the list L, if there is no previously existing open path between them. In this example, the list L equals [{C,D}, {D,E}, {A,D}, {A,E}, {A,C}] at the end of the phase as these pairs of nodes are not directly connected in the first phase but they do have the mutual information greater than ε. The graph generated at the end of this step is shown in Fig. 3.4.(b).

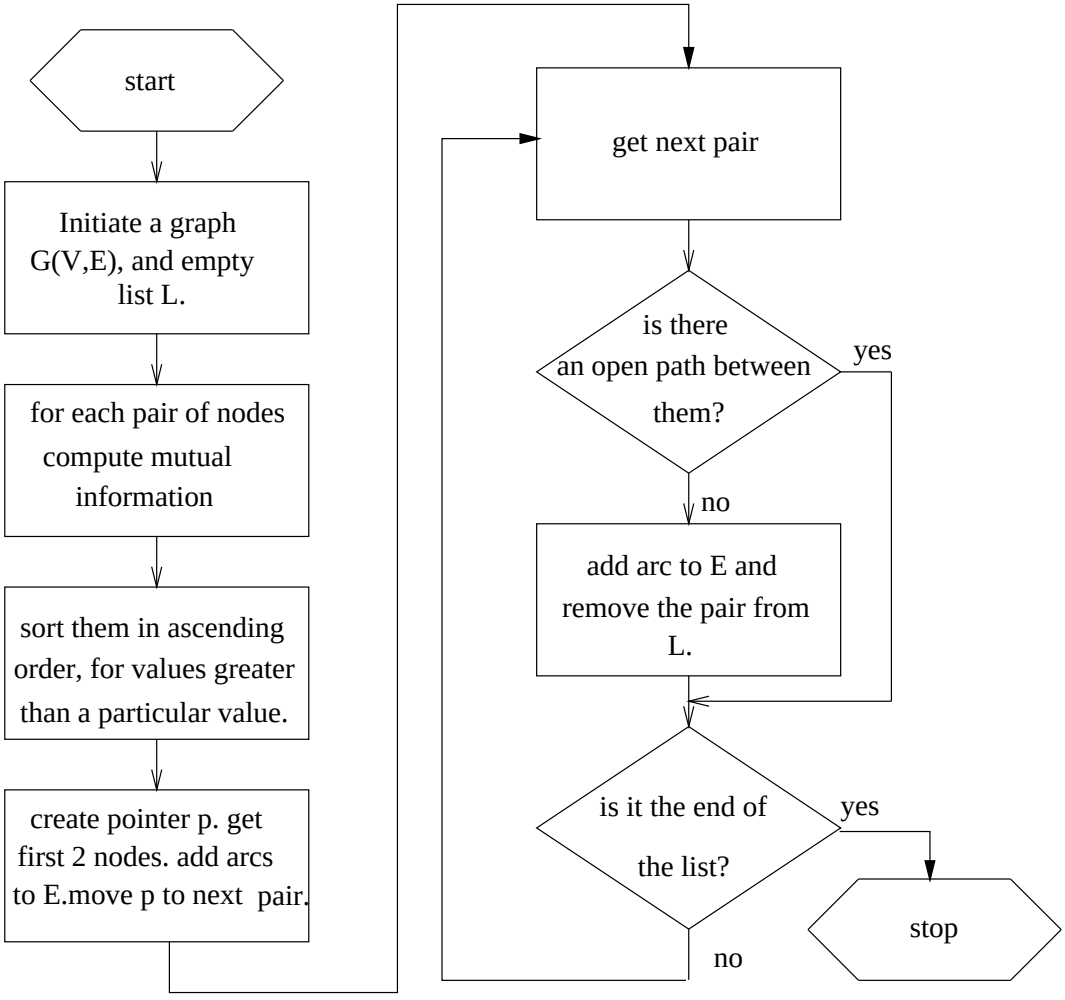

Figure 3.3. Step 1: Drafting.

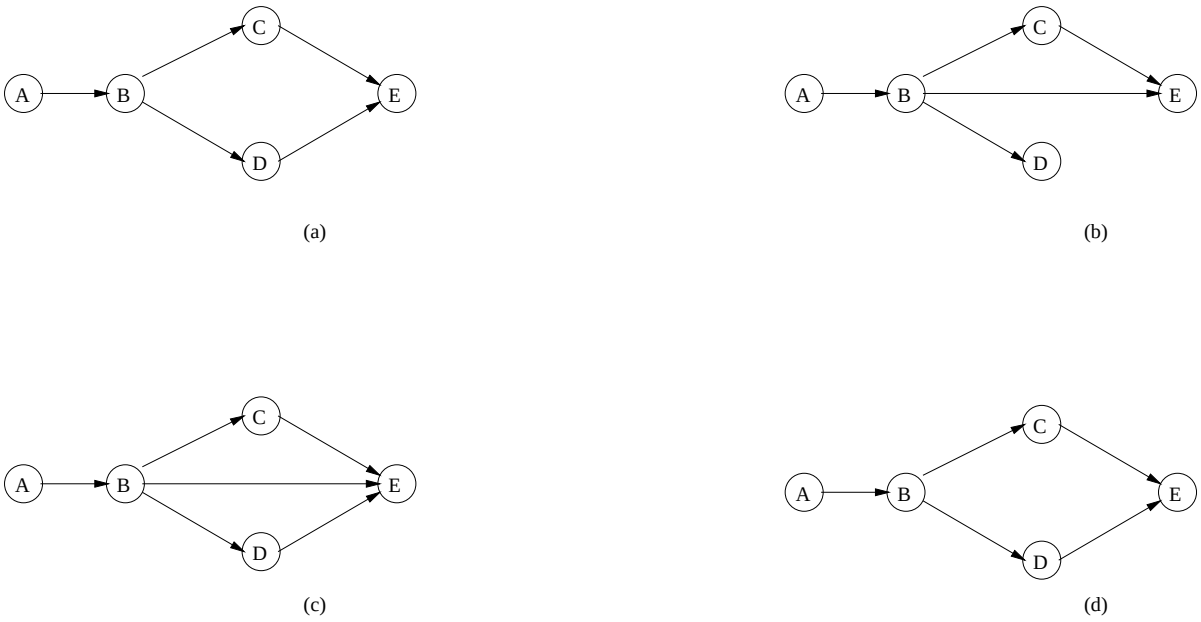

Figure 3.4. Working Mechanism.

This step of the process tries to model the graph as close to the final model as possible using only pair-wise mutual information tests, that is no conditional independence tests are involved. Though the graph obtained from here can be anywhere from an empty graph to the exact final result, the efficiency is improved if the graph is close to the final model. When searching the draft, we try to minimize the number of missing or wrongly added arcs. A stop condition is used to minimise these errors. This drafting step ends when every pair-wise dependency is expressed by an open path in the draft. This step is essentially the Chow-Liu algorithm and it guarentees the constructed network is the same as the final one. Therefore this algorithm can be viewd as a special case for the Bayesian networks that have tree structures.

3.3.2 Step 2:Thickening

The steps here are again easily explained with the help of another flow chart.

In this step CI tests and d-separation analysis are used to find out if the pair of nodes that were left out in step 1 because they already had some open paths, should be connected or not. We cannot be sure if the connection is really needed, but we can know that no arcs will be missed at the end of this step as we use only one CI test to make a desicion. The procedure find-cut-set tries to get a cut-set that can

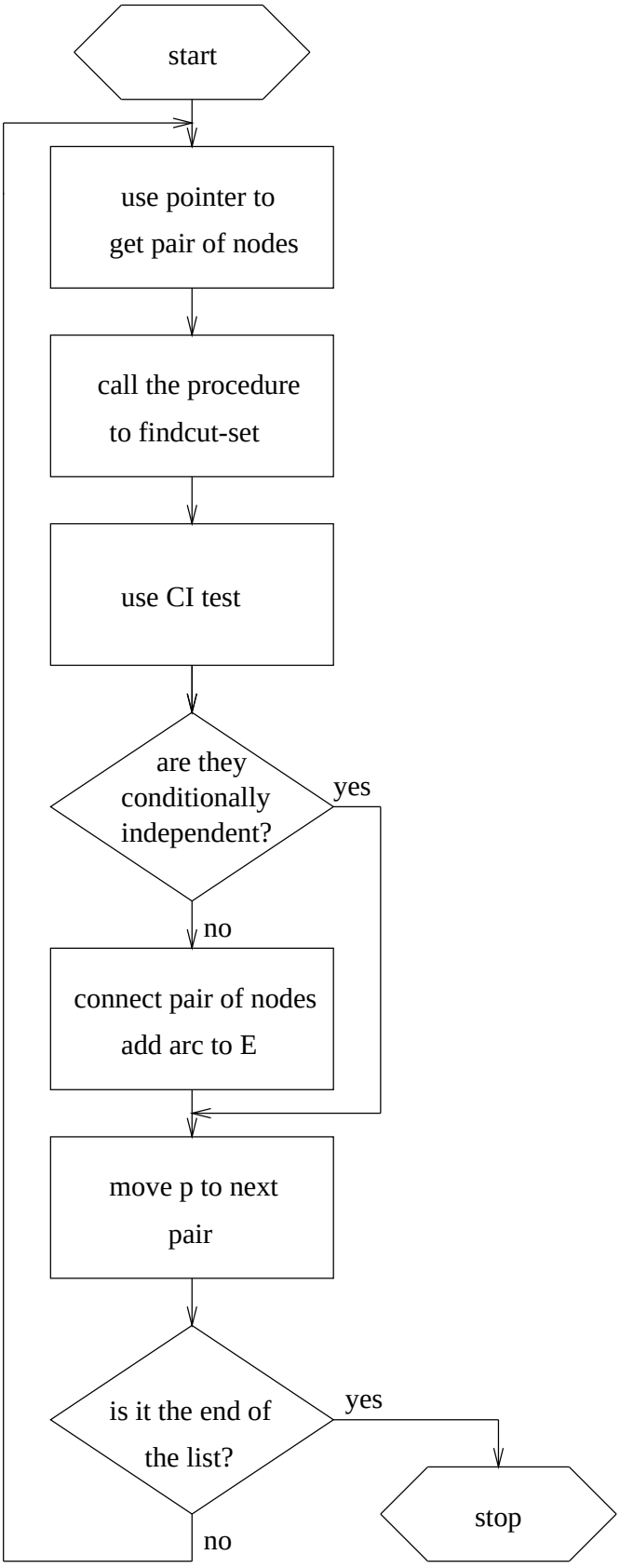

Figure 3.5. Step 2: Thickening.

d-seperate the two nodes, and this is done for every pair of nodes. Then it uses a CI test to see if the two nodes are independent conditional on the cut-set. The arc is added if the nodes are not conditionally independent. Some arcs can be wrongly added, because some needed arcs may be missing and they hinder finding a proper cut-set.

The graph after the second step is shown in Fig. 3.4.(c). Arc (D,E) is added because D and E are not independent given B, which is the smallest cut-set between D and E. Also, arc (A,C) is not added because the conditional independence tests show that A and C are independent given B. Similarly, for the same reasons we do not add arcs (A,D), (C,D), (A,E). From the grpah we can also see that after this stpe the graph obtained is an I-map of the underlying model.

3.3.3 Step 3: Thinning

For the two nodes under consideration, if there are other paths, other than the arc, then the arc is temporarily removed from E, and the procedure is called again to find a cut-set that can d-seperate the two nodes. Now given tha cut-set the CI test is done to find out whether the two nodes are conditionally independent. If so, the arc is permanently removed or else it is added back. This step is basically to find the arcs which may be been wrongly added in the previous steps. Here we also make sure all the addes arcs are needed and there are no extra ones, by using only one CI test. Since the current graph had become an I-map at the end of the previous step, we can be sure that the descion to remove arcs is correct. In this step, we try to find that an arc connecting a pair of nodes are also connected by other paths. This is done because it is possible that the dependency between pair of nodes could not be due to the direct arc. Then the algorithm removes this arc and uses the procedure to find the cut-set. Then the conditional independence test is used to check if the two nodes are independent condtional on the cut-set. If so, the arc is permanently removed as the two nodes are independent, or else the arc is plced back since they are conditonally dependent. This is repeated till all the arcs are examined.

This step is shown in Fig. 3.4.(d) for the example, and it has the final complete graph. Here we can see edge (B,E) is removed permanently because B and E are Independent given C,D. So at the end of this third step the correct Bayesian network is obtained.

3.3.4 Finding Minimum Cut-Sets

In the steps 1 and 2 we call the procedure find-cut-set to get a cut-set between the two nodes and then use this cut-set in the conditional independence test. [19] also proposes an algorithm to find minimum cut-sets, which proves that a minimum cut-set between two nodes can be always found.

```
      Begin
       input node1, node2;
       Explore(listOfPath, node1, node2);
       Store(ListOfPath, OpenPath, Closepath);
        Do
         While there are open paths with only one node Do
   put these nodes of each such path in cut-set;
   take off all the blocked paths by these nodes from the open and closed path set;
           Find paths, from the closed path set, opened by the nodes
           in the block set and move them to the open path set;
   Remove the nodes that are also in the cut-set to shorten such paths;
End While
         If there are open paths Do
    In the cut-set find and put the node that can block the max number of remaining paths;
    Remove all the blocked paths by the node from both the open and closed path set;
    Find paths, from the closed path set, opened by this node and
    move them to the open path set;
    Remove the nodes that are also in the cut-set to shorten such paths;
         End If
      Until there is no open path
    End
```

We have two sets namely the open and closed path set. We find first all the adjacency paths between the two nodes and and put the paths into the two sets. In the cut-set we put the non-converging paths

that are connected to both the nodes. This is done because these have to be in every valid cut-set. This is repeated to find a node that can block the maximum number of paths. Put this in the cut-set and repeat till all the open paths are blocked.

3.3.5 Complexity Analysis

In general, most of the running time of learning alorithms is consumed by data retrieval from databases. Suppose a data set has N attributes, the maximum number of possible values of any attribute is r, and an attribute has a maximum of k parents. The computation of mutual information requires at the most $O(r^2)$ basic operations and computaion of conditional mutual information requires at the most $O(r^{k+2})$ basic operations, since the condition set has k nodes at the most. So, the conditional mutual information test has a complexity of $O(r^N)$. The first step has a complexity of $O(N \log N)$ steps, for sorting the mutual information of pairs. The procedure has $O(N)$ basic operations. So as a whole with respect to the CI tests, Step 1 needs $O(N^2)$ mutual information computations and Step 2 needs at most $O(N^2)$ number of CI tests. Similarly Step 3 requires at most $O(N^2)$ number of CI tests, because a descision requires one CI test. So as a whole the algorithm requires $O(N^2)$ CI tests in the worst case.

CHAPTER 4

SAMPLING

Bayesian networks are the best suited to represent joint probability distributions, as they include the conditional independence relationships among the variables in the network. An attractive feature of Bayesian network is that its amenability to recursive and incremental computation of schemes. Let H denote an hypothesis, $e_n = e^1, e^2, ...e^n$ denote a sequence of data observed in the past, and e denote a new fact. The original way to calculate the belief in H, $P(H \mid e_n, e)$ would be to add the new data e to the past data e_n and perform a overall computation of the effect on H of the entire data set $e_{n+1} = e_n, e$. This computation is time and memory intensive because the computation of $P(H \mid e_n, e)$ becomes more and more complex as the size of the set increases. This calculation can be simplified by discarding the past data once we have computed $P(H \mid e_n)$. Now we can calculate the value of the new data by Eq. 4.1

$$P(H \mid e_n, e) = P(H \mid e_n) \frac{P(e \mid e_n, H)}{P(e \mid e_n)} \tag{4.1}$$

Also, we know from Eq. 3.2

$$P(H \mid e) = \frac{P(e \mid H)P(H)}{P(e)} \tag{4.2}$$

Comparing Eq. 4.1 and Eq. 4.2 we can see that the old belief $P(H \mid e_n)$ assumes the role of prior probability in the computation of the new impact. It can also be seen that it completely summarizes the past experiences and if we need updating, we need to multiply only by that likelihood function $P(e \mid e_n, H)$, which as we can see given the hypothesis and past observations it measures the probability of the new data e. Likelihood is the hypothetical probability that an event that has already occured would yield a specific outcome. The concept differs from that of a probability in that a probability refers to the occurence of future events, while a likelihood refers to past events with known outcomes. This recursive formulation still would be cumbersome but for the fact that the likelihood function

is often independent of the past data and involves only e and H. Using the condtional independence condition we can write,

$$P(e \mid e_n, H) = P(e \mid H) \;\; and \;\; P(e \mid e_n, \neg H) = P(e \mid \neg H) \tag{4.3}$$

and dividing Eq. 4.1 by the complementary equation for $\neg H$, we can write

$$O(H \mid e_{n+1}) = O(H \mid e_n) L(e \mid H) \tag{4.4}$$

If we multipy the current posterior odds $O(H \mid e_n)$ by the likehood ratio of e, upon the arrival of each new data e, as shown in Eq. 4.4, shows a simple recursive procedure for updating the posterior odds. Odds is the probability that the even will occur divided by the probability that the event will not occur. $O(H \mid e_n)$ is the prior odds relative to the next observation, while $O(H)$ is the posterior odds that has evolved from the previous observation not included in e_n. Taking the logarithm of Eq. 4.4,

$$\log O(H \mid e_n) = \log O(H \mid e_n) + \log L(e \mid H) \tag{4.5}$$

The simplicity of the log-likelihood calculation had led to a variety of applications, especially in intelligence gathering tasks. For each new report, we can estimate the likelihood ratio L, which can be easily incorporated in the already accumulated overall belief in H.

4.1 Causal Networks

The important requirement of Bayesian network is that it incorporates d-separation properties of the domain modeled and it need not reflect cause-effect relations. But there is, however, a good reason to strive for causal networks. Causation is one of the basic primitives of probability because it is an indispensible tool for structuring and specifying probabilistic knowledge. Since the semantics of causal relationships are preserved by the syntax of probabilistic manipulations no auxillary devices are needed to force conclusions. For example if I figured out that the cause of my slipping was due to a wet pavement, I could no longer consider other events as the wetness of the pavement is confirmed. The facts that it rained that day or the sprinkler was on or my friend also slipped and broke a leg should no longer be considered once the wetness of the pavement is identified as the direct cause of the

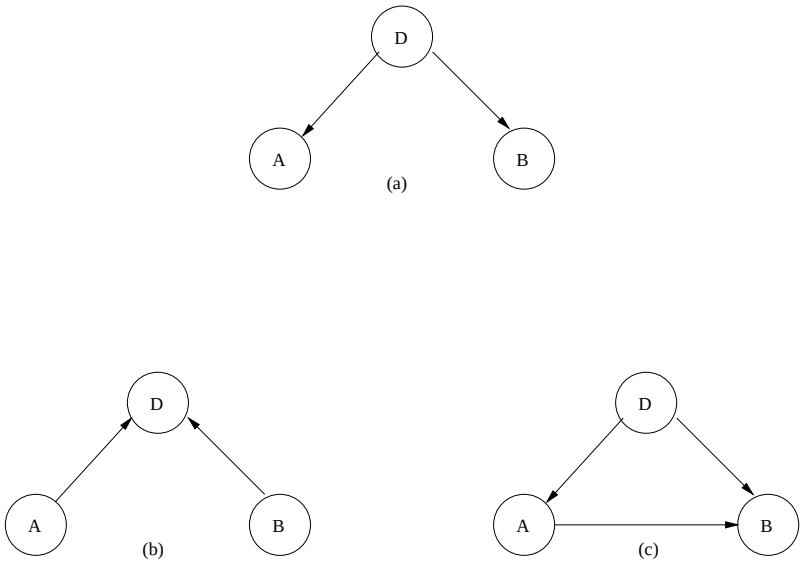

Figure 4.1. Causal Diagram.

accident. The asymmetry conveyed by the causal directionality can be used for encoding more complex and intricate patterns of relationships. It can now be understood that once a consequence is observed its causes can no longer remain independent because confirming one cause lowers the likelihood of the other. Causal directionality conveys the message that two events do not become relevent to each other merely by virtue of predicting a common consequence, but they do become relevant when the consequence is actually observed. The opposite is true for two consequences of a common cause, typically the two become independent upon learning the cause, as discussed when learning the structure from data.

Consider the Fig 4.1., which represents a disease D and two tests A and B. It is usually represented by the physician as shown in Fig 4.1. (b), generally as opposed to the model Fig 4.1. (a). But it is not correct to represent the situation as done in Fig 4.1. (b) because the two tests are independent, and to correct the model, we must introduce an extra link as shown in Fig 4.1. (c). Now the structure is not minimal and to get a correct model, it is not sufficient to acquire $P(D \mid A, B)$ together with P(A) and P(B). The conditional probabilities for Fig 4.1. (a) reflect the general properties of the relation between diseases and tests and they are the ones that a manufacturer of tests can publish, whereas the conditional probabilities for Fig 4.1. (b) are a mixture of disease-test relations and prior frequencies of the disease. The presence of interventions provide another good reason for the use of causal networks.

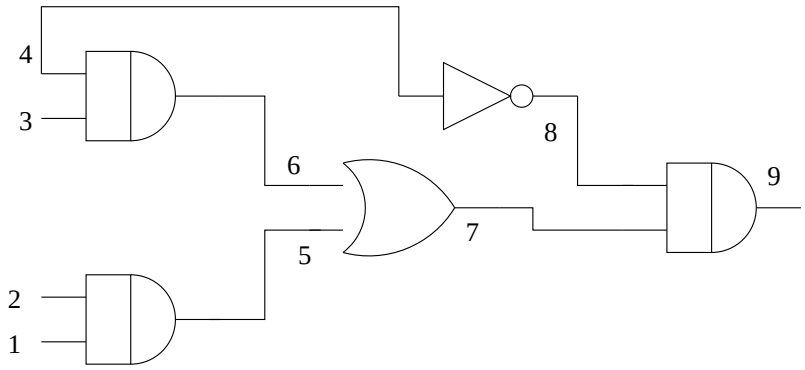

Figure 4.2. A Small Circuit.

An intervention is an action that has an impact on the state of particular variables. So in causal models the impact of the intervention will spread in the causal direction and not in the opposite way. If the model does not reflect causal directions it cannot be used to simulate the impact of interventions.

As defined before, Bayesian networks are directed acyclic graphs in which each node represents a uncertain quantity or random variable, which can take two or more values. The arcs which connect the nodes signify the existence of direct causal influences between the linked nodes or variables and the strengths of these influences are quantified by the conditional probabilities. The advantage of network representation is that it is easy to express directly the fundamental qualitative representation of direct dependencies. Also, it can display a consistent set of additional direct and indirect dependencies and preserves it as a stable part of the model, independent of the numerical estimates. The directionality of the arrows is essential to display nontransitive dependencies that is if two causes are independent to each other, they are still relevant to the effect and induced dependencies, that is once a consequence is observed, its causes can no longer remain independent, because one cause lowers the likelihood of the other. If the arcs are stripped of their directions some of the relationships would be misinterpreted. Thus by diplaying irrelevances in the domain modelled, the causal schemata minimizes the number of relationships that need to be considered while a model is constructed and this in effect legitimizes many future local references.

A Markov network is an undirected graph whose links represent symmetrical probabilistic, while a bayesian network is a directed acyclic graph whose arrows represent causal influences. The main weakness of Markov networks is their inability to model in the structure the induced and non-transitive dependencies. In this representation any two independent variables will be directly connected by an

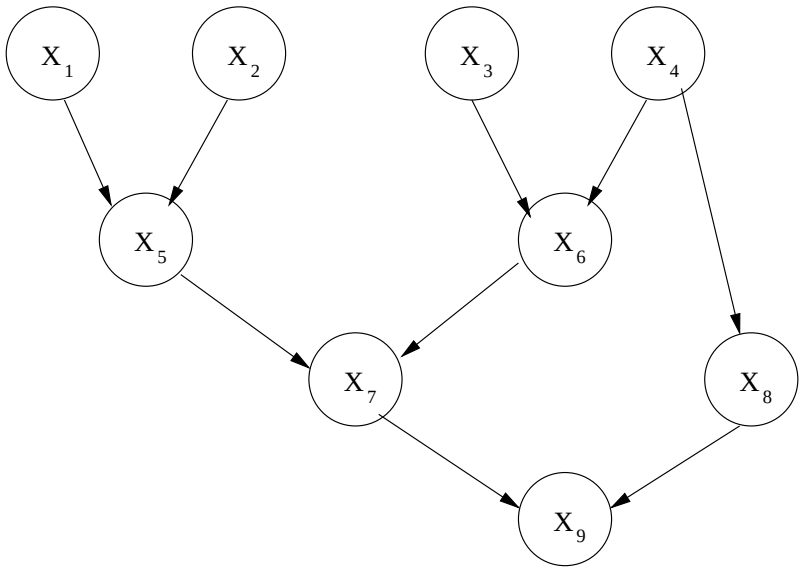

Figure 4.3. BN Corresponding to the Circuit.

edge, merely because someother variable depends on both. Due to this method of modelling, many useful independencies go unrepresented in the network, thus making it long and complex. Bayesian networks efficiently overcomes this deficiency by using a much richer language of directed graphs. So these directions of arrows help us to distinguish genuine dependencies from spurious dependencies induced by hypothetical observations. The concept of d-separation as explained in Section 3.2, is efficently used to address these dependencies. An even bigger advantage of directed graphical representaions is that, they make it easy to quantify links with local, conceptually meaningful parameters that turn the network as a whole into globally consistent knowledge base.

From these discussions it is clear that causal representation is best suited for representation of various phenomenons. Thus it is no problem to configure a graph which represents phenomena with explicit notions of neighborhood or adjacency like electronic circuits and communication networks, as discussed in [10]. The determination of direction of the network to be modelled becomes the imporatant task in causal networks. In modelling logic circuits, the direction of flow is known and it becomes much easier to construct a causal network. So Bayesian networks are the best suited for the representation of VLSI circuits, than undirected graphs like Markov networks.

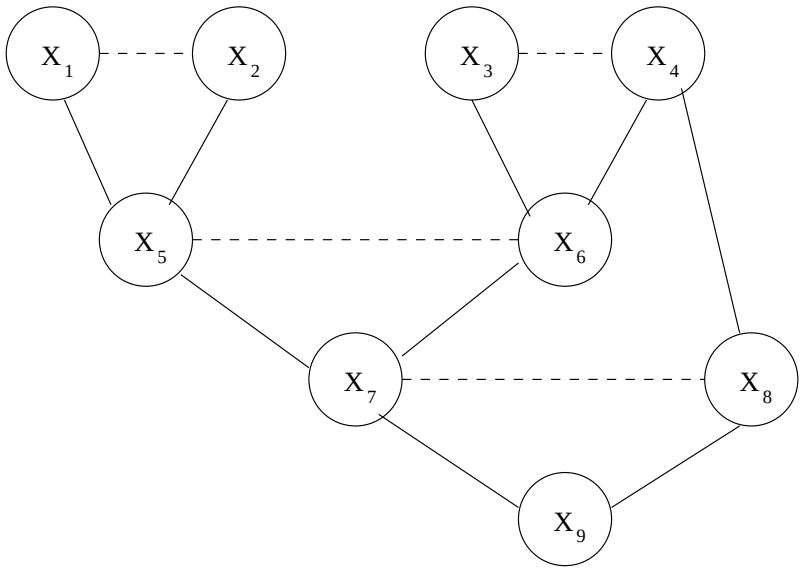

Figure 4.4. Moral Graph.

4.2 Inference

The Bayesian network not only models causality but also makes inference much easier. The inference done here is split into many steps. First convert the network into a junction tree of cliques and then use probabilistic logic sampling to acheive our purpose. The steps involve the formation of a moral graph, and the process is called compilation, then the moral graph is triangulated. The clique set is identified and the junction tree of cliques is formed.

4.2.1 Moral Graph

Given a Bayesian network, a moral graph is obtained by 'marrying parents', that is adding undirected edges between the parents of a common child node. Before this step, all the directions in the DAG are removed. This step ensures that every parent child set is a complete subgraph. These undirected links are called the moral links. The moral graph represents the Markov structure of the underlying joint function. Some of the independencies displayed in the original DAG structure will not be graphically visible in the moral graph as it is undirected and has additional links added to it. Some of the independencies that are lost in the transformation contributes to the increased computational efficiency, but does not affect the accuracy. The moral graph of the Fig. 4.3. is shown in Fig. 4.4.

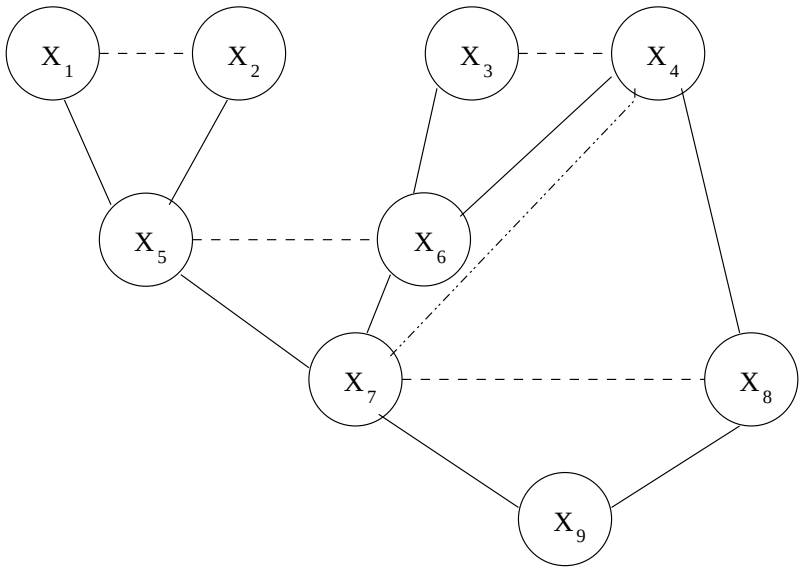

Figure 4.5. Triangulated Graph.

In the figure it seen that the direction of the edges have been removed and the parents of a common child have been 'married', that is a link is added between them. The added links are shown as dotted lines and we can see that they connect X_1 and X_2, X_3 and X_4, X_5 and X_6 and between X_7 and X_8. If the underlying DAG structure is denoted as D, then the moral graph is denoted as D^m.

4.2.2 Triangulation

The moral graph is said to be triangulated if it is chordal. The undirected graph G is called chordal or triangulated if every on of its cycles of length greater than or equal to 4 posesses a chord [9], that is we add additional links to the moral graph, so that cycles longer than 3 nodes are broken into cycles of three nodes.

In the Fig. 4.5. we have added a dash-dotted line between X_4 and X_7 to triangulate the moral graph. The easiest way maybe to add all possible links between nodes to make the graph chordal, also transforming it into a complete graph. But the challenge is to use minimal possible additional links, which is in general a NP-hard problem and various heuristics are used. In HUGIN, a minimum fill-in edges heuristics is used as explained in [9].

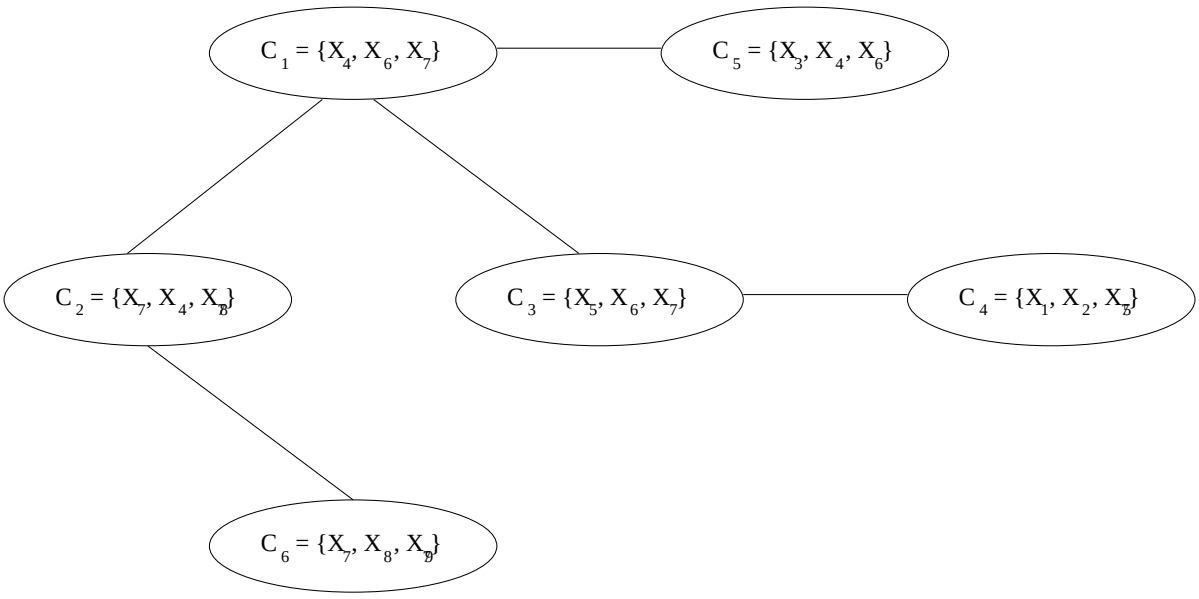

Figure 4.6. Junction Tree.

4.2.3 Junction Tree

The juntion tree is defined as a tree with nodes representing cliques (collection of completely connected nodes) and between C_i and C_j in the tree T there is a unique path. The running intersection property states that 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 of the junction tree is used in the local message passing. The construction of the junction tree breaks down to the formation of chordal graph from the moral graph, where the set of potential cliques is created, referred to as the elimination set. Then the elimination set of cliques is reduced and tree links between the cliques are added forming the junction tree.

In constructing the chordal graph and elimination set from the moral graph, first all the vertices of the moral graph are unnumbered. The vertex that needs the minimum number of additional edges between its neighbors is chosen and is given the highes available node number, say i, starting from the number equal to the total number of nodes. Then the set C_i is formed of the selected vertex and its still unnumbered neighbors. Edges are then filled in between any two unlinked nodes in this set and the node number is decremented by 1. This process is repeated till there are no unnumbered nodes. The resultant graph is chordal and the cliques C_i's are termed the elimination set of the graph, subset of which is used to construct the junction tree.

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.

The resultant elimination set $\{C_i\}$ (which is the superset of the clique-set) obtained from the running example are

$$\begin{aligned}\{C_9, \cdots, C_1\} &= \{\{X_9, X_8, X_7\}, \{X_3, X_4, X_6\}, \{X_2, X_1, X_5\}, \{X_1\}, \{X_5, X_6, X_7\}, \\ &\quad \{X_8, X_7, X_4\}, \{X_7, X_6, X_4\}, \{X_6\}, \{X_4\}\}\end{aligned}$$

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 4.6. shows the junction tree for the example considered. 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. For our example in Figure 4.6., 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. The separator set is the non-null set of nodes common between every clique with an edge between them. These separator sets are used for 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.

4.2.4 Propagation in Junction Trees

After the junction tree of cliques is formed, the next step is to form the distribution function of cliques. It has been proved in [9] that the dependency properties of a DAG, which carry over to moral graphs, are also preserved in the triangulated moral graph. Also, the joint probability function that factorizes on the moral graph will also do so on the triangulated on since each clique in the moral graph is either a clique in this graph or a 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 between the adjacent cliques in the junction tree. The joint probability function factorizes over the junction tree in the following form [9]:

$$p(x_1,\cdots,x_N) = \prod_{c\in \mathrm{CS}} p(\mathbf{x_c}) / \prod_{s\in \mathrm{SS}} p(\mathbf{x_s}) \tag{4.6}$$

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:

$$P(x_1,\cdots,x_N) = \prod_{c\in \mathrm{CS}} \phi_c(\mathbf{x_c}) \tag{4.7}$$

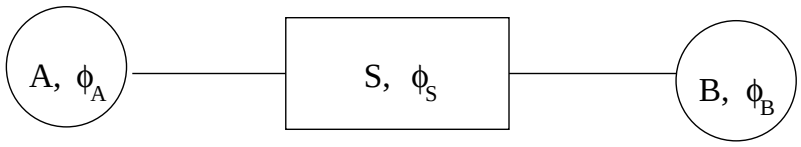

Figure 4.7. Two Cliques with the Separator Set.

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 two cliques A and B have probability potentials ϕ_A and ϕ_B, respectively and S be the set of nodes that separates A and B as shown in Fig 4.7. When there is new evidence for some node, it will change the probabilities of all the other nodes such that the neigboring cliques agree on the probabilities of S, the separator set. 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. 4.9. To achieve this we need to transmit 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 and hence parallel message passing is possible.

$$\phi_S^* = \sum_{X \in A, X \notin S} \phi_A \tag{4.8}$$

$$\phi_B^* = \phi_B \frac{\phi_S^*}{\phi_S} \tag{4.9}$$

However we need not initiate the message passing over the whole network for every new evidence as there is a two phase message passing scheme that can integrate all new evidence in two passes. A clique is selected to be the root node, then all the leaf nodes send messages towards the root node, which are re-computed using Eqs. 4.8 and 4.9 at each node. Then as the messages from all the leaf

nodes are received, they 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.

4.2.5 Probabilistic Logic Sampling

Probabilistic Logic Sampling is a method proposed by Henrion [56] which employs a stochastic simulation approach to make probabilistic inferences in large multiply connected networks. If we represent a Bayesian network by a sample of m deterministic scenarios s=1, 2,.....m and $L_s(x)$ is the truth of event x in scenario s, then uncertainty about x can be represented by a logic sample, that is the the vector of truth values for the sample of scenarios:

$$L_x \equiv [L_1(x), L_2(x), ...L_m(x)]. \tag{4.10}$$

If we have the prior probability p_x, we can use a random number generator to produce a logic sample for x. This method of sampling proceeds as explained below, given a Bayesian network with priors specified for all source variables and conditional distributions for all others:

1. Use a random number generator to produce a sample value for each root node in the network and a sample implication rule for each parameter of each conditional distribution, using the corresponding probabilities.

2. Proceed down through the network following the direction of the arrows from the root nodes using simple logical operations to obtain the truth of each variable from its parents and the implication rules.

3. Repeat steps 2 and 3 m times to obtain a logic sample for each variable.

4. Estimate prior marginal probability of any event by computing the number of times in which they are true.

5. Estimate posterior probability on any of the already computed variable as the fraction of the number of times the event occurs out of the possible scenarios in which the given condition is true.

We stop with step 3 as we need only the logic sample and we do not compute the posterior probabilites. We use this logic sampling method inside each clique of the junction tree. To start with, the first clique is considered and the sampling is done inside to arrive at the samples for the nodes inside the clique. Once values are fixed for these nodes, they are propagated to the next clique with the help of the separator set. As the nodes of the separator set are in both the cliques, one or more of the nodes are instantiated. With these values the rest of the nodes are sampled with logic sampling and this goes on till the end of the junction tree.

CHAPTER 5

EXPERIMENTAL RESULTS

This model proves to be an efficient technique for power estimation by providing a minimum number of samples. As said before the method bridges the gap between simulative and probabilistic techniques. This is acheived by an effecient model of Bayesian network, by learning the structure from the input data. The learned structure of the Bayesian network represents the graphical structure of the input data. The learned structure can now be used to generate any number of vectors. The generation of the vector set is done efficiently by exploiting the sampling techniques applied on Bayesian networks. Sampling is done to derive vectors in any needed compaction ratio. The reduced vector set is then fed into HSPICE to get an accurate estimate of the average power. The power estimate is then compared with the value got from simulating the large vector set. If a is the original length of the vector and b is the compacted length, then a/b is called the compaction ratio (CR). We have acheived very high compaction ratios of upto 300 which have never been acheived before.

We have fixed the larger vector set to be 60,000 highly-correlated input patterns generated by simulating the various benchmark circuits. The vectors were then taken from random output nodes of these circuits. This is done to induce a high correlation between the vectors so that they are as close as possible to the real world data. Also different benchmark circuits were used for generating these vectors, for example to obtain vectors for C499 the circuit C3540 was simulated. The correlation between the vectors can be seen in the constructed Bayesian network.

The overall process flow is as shown in Fig. 5.1.. As shown, we first generate the the large vector set. Then the power consumed by each benchmark circuit is calculated using the large vector set as the input stimuli. This value is used as the basis for comparison with the power estimate generated from the compacted vector set.

Then the second step is to use this vector set to learn the Bayesian network. We use Power Constructor [16] to acheive our purpose. The power constructor is an efficient tool which provides an accurate

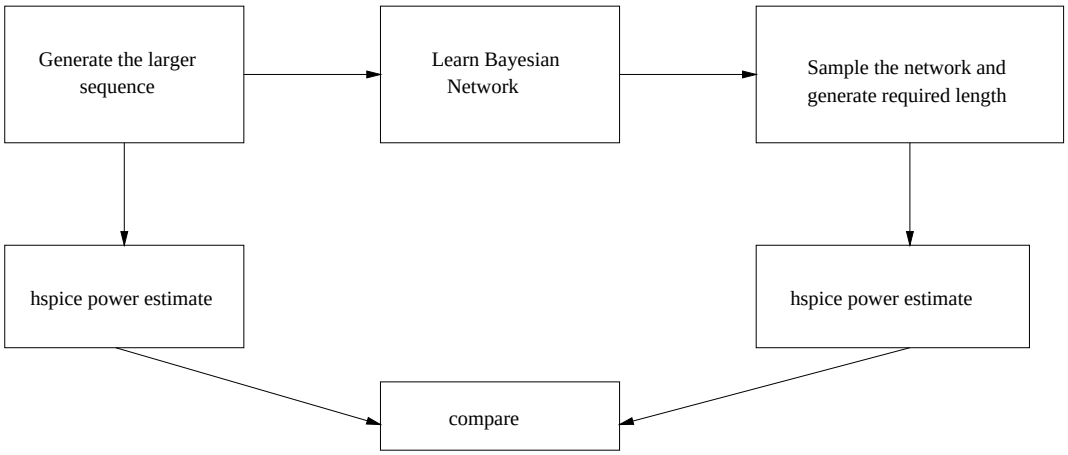

Figure 5.1. Process Flow.

Bayesian network in the HUGIN [23] readable format. The algorithm used here takes a database as input and constructs the belief network structure as output. This algorithm extends the Chow and Liu's tree construction algorithm to belief network construction by using a three phase mechanism. The three phases are namely drafting, thickening and thinning. In the first phase, this algorithm computes the mutual information of each pair of nodes as a measure of closeness and creates a draft based on this information. In the second phase, the algorithm adds edges when the pair of nodes that cannot be d-separated. The third phase, each edge of the the current graph is examined using CI tests and will be removed if the two nodes of the edge can be d-separated. The algorithm guarentees that the perfect map of the underlying model is generated. The process is mounted on a 32-bit windows systems on PC and can be run on Windows 9x and XP versions. Since the construction engine is an ActiveX DLL, it can be integrated into other belief network, data mining or knowledge base systems. Also, modified conditional independence test method is used to make the results more reliable when the data set is not large enough.

After constructing the Bayesian network, the next step as shown in the process flow graph is the generation of a compacted vector sequence. It is done by sampling the belief network using a code written in HUGIN API. The functions in Hugin are given in terms of ISO/ANSI C function prototypes. It contains a high performance inference engine that can be used as the core of knowledge based systems built using Bayesian belief networks or influence diagrams. With the probabilistic descriptions of

Table 5.1. Power Estimates of ISCAS '85 Benchmark Suite.

Circuit	No of Inputs	60K	Compaction Ratio			
			40	80	200	300
C17	5	2.9973	2.9826	2.9889	2.9829	2.9799
C432	36	4.1304	4.1669	4.1980	4.1474	4.1545
C499	41	3.0005	2.9972	3.0177	3.0927	3.1116
C1355	41	3.2257	3.2348	3.2478	3.3319	3.3245
C1908	33	5.3925	5.3634	5.4259	5.4095	5.3511
C3540	50	1.5969	1.5875	1.5866	1.5499	1.5769
C6288	32	2.2521	2.3112	2.3116	2.3142	2.3197
		Avg % Error	0.77	1.02	1.83	1.86

Table 5.2. Error Estimates in Power Estimation.

Circuit	No of Inputs	Percentage Error for CR			
		40	80	200	300
C17	5	0.49	0.44	0.48	0.58
C432	36	0.88	1.6	0.41	0.58
C499	41	0.10	0.57	3.07	3.7
C1355	41	0.28	0.68	3.2	3.0
C1908	33	0.50	0.61	0.31	0.76
C3540	50	0.58	0.64	2.9	1.25
C6288	32	2.6	2.6	2.7	3.0
	Avg % Error	0.77	1.02	1.83	1.86

causal relationships in the domain, one can build knowledge bases that model the application domain. Given this description, fast and accurate reasoning can be done with the inference engine.

Before the network can be used for inference, it must be compiled. During compilation, first the links to decision nodes are removed. Then the network is converted into its moral graph, that is the parents of each node are ”married” and the directions of the edges are dropped and the utility nodes are removed. Then the moral graph is triangulated, that is fill-in links are added so that each cycle of length greater than three has a chord (an edge connecting two non-consecutive nodes). Then the cliques (maximal complete sets) of the triangulated graph are identified, and the collection of cliques is organised as a tree, with the cliques forming the vertices of the tree. Such a tree is called a junction tree. If the original tree is disconnected, there will be a tree for each connected component. Lastly potentials are associated with the cliques and the links of each junction tree. These potentials are initialized from the conditional probability tables, using a sum propagation. Vectors of values over the set of variables in the network (sampling) can be generated with respect to the conditional distribution. If the domain is compiled sampling the configuration with respect to the current distribution on the junction tree can be done. Since the distribution is an influence diagram, all the decisions must be instantiated and propagated. The network must be an acyclic directed graph and is true in our case. Also, all chance nodes must have a valid conditional probability distributions, which is also true in our case, and then the decisions are instantiated.

Table 5.1. shows the power estimates of the ISCAS ’85 benchmark suite. The second column shows the number of inputs in each each circuit and the Bayesian network model has the number of nodes equal to the number of inputs. So the largest Bayesian network used was 50 for C3540 and the smallest is of 5 nodes for C17. Table 5.2. shows the percentage error obtained for each circuit and for each compaction ratio. It is seen that as the compaction ratio increases the error% increases, but it is not that significant. Also, the error was only 0.77% for CR 40 with a maximum error of 1.86% for CR 300, with the highest error of 3.7% for C499 for CR 300. Circuit C1908 is seen to have the lowest error% for all compaction ratios and C6288 having high errors for most CR’s.

The following tables show the crosstalk estimation values for the various benchmark circuits for both the bigger vector set and the compacted one with the compaction ratio of 40. These values denote the joint probability of the two nodes in consideration. HSPICE is used to print the voltages values for

Table 5.3. Joint Probability Switching Estimate for Nodes 300 and 330 of C432 for 60K.

Node 300	Node 330			
State	00	01	10	11
00	0.06	0.19	0.19	0.54
01	0	0	0	0
10	0	0	0	0
11	0	0	0	0

Table 5.4. Joint Probability Switching Estimate for Nodes 300 and 330 of C432 for CR 40.

Node 300	Node 330			
State	00	01	10	11
00	0.053	0.19	0.19	0.56
01	0	0	0	0
10	0	0	0	0
11	0	0	0	0

the particular input vectors of the two nodes, which are then changed to their switching values. From these voltage values the joint probability is calculated by finding the number of occurences. The main idea of this part of the result is to estimate the worst-case crosstalk, but the table also gives an estimate of the probability of leakage occuring in the nodes. We have limited to only a few nodes due to high running time taken by HSPICE, otherwise any number or all of the nodes can be estimated.

The tables 5.3. and 5.4. were plotted for the nodes 300 and 330 which are the inputs to a gate selected at random. The table 5.4.shows that the compacted vector set gave the similar crosstalk estimation results as the original vector set used in 5.3., proving that the smaller vector set is equally efficient in modelling crosstalk. The simultaneous switching noise of one node switching from 0 to 1 and the other switching 1 to 0 at the same time yields the maximum crosstalk. Here we can see that the particular set of input used has zero or very low worst case switching noise. Also we can see that the leakage is pretty high, and so the designers should pay more attention to reducing the leakage.

The two tables 5.5. and 5.6. are for the circuit C499, considering the nodes 557 and 558. We can see here too that the compacted vector set yielded similar values with respect to the result from the larger vector set. For this set of inputs the coupling noise is as significant as the leakage. The worst-case crosstalk probability is .07 for node 557 switching from 0 to 1 and node 558 switching from 1 to 0 and .07 for 557 switching from 1 to 0 and 558 switching from 0 to 1.

Table 5.5. Joint Probability Switching Estimate for Nodes 557 and 558 of C499 for 60K.

Node 557	Node 558			
State	00	01	10	11
00	0.054	0.058	0.059	0.057
01	0.066	0.06	0.07	0.066
10	0.06	0.071	0.063	0.065
11	0.065	0.062	0.068	0.07

Table 5.6. Joint Probability Switching Estimate for Nodes 557 and 558 of C499 for CR 40.

Node 557	Node 558			
State	00	01	10	11
00	0.056	0.057	0.059	0.057
01	0.064	0.06	0.07	0.064
10	0.06	0.07	0.063	0.067
11	0.066	0.06	0.068	0.07

Table 5.7. Joint Probability Switching Estimate for Nodes 141 and 113 of C1355 for 60K.

Node 141	Node 113			
State	00	01	10	11
00	0.25	0	0	0
01	0	0.24	0	0
10	0	0	0.25	0
11	0	0	0	0.24

Table 5.8. Joint Probability Switching Estimate for Nodes 141 and 113 of C1355 for CR 40.

Node 141	Node 113			
State	00	01	10	11
00	0.25	0	0	0
01	0	0.24	0	0
10	0	0	0.25	0
11	0	0	0	0.24

Table 5.9. Joint Probability Switching Estimate for Nodes 2427 and 2340 of C1908 for 60K.

Node 2427	Node 2340			
State	00	01	10	11
00	0.19	0.02	0.02	0
01	0.05	0.17	0	0.02
10	0.05	0	0.16	0.02
11	0.01	0.04	0.04	0.14

Table 5.10. Joint Probability Switching Estimate for Nodes 2427 and 2340 of C1908 for CR 40.

Node 2427	Node 2340			
State	00	01	10	11
00	0.19	0.02	0.02	0
01	0.06	0.2	0	0.02
10	0.06	0	0.16	0.02
11	0.01	0.04	0.04	0.15

The tables 5.7. and 5.8. represents the joint probablity distribution of nodes 141 and 113 which are the two inputs of a NAND gate. The particular input set shows that there is a high probability of leakage when both the nodes are switching from 0 to 0 and from 1 to 1.

The tables 5.9. and 5.10. captures the nodes 2427 and 2340 which are inputs of a NOR gate. Here we can see a high probability of leakage than crosstalk noise as both nodes switching from 0 to 0 and 1 to 1 have a high joint probability.

The nodes considered are 4899 and 4925 in the tables 5.11. and 5.12.. Here we can see that the switching from 1 to 1 in both the nodes has the highest probability, showing high probability of leakage.

Table 5.11. Joint Probability Switching Estimate for Nodes 4899 and 4925 of C3540 for 60K.

Node 4899	Node 4925			
State	00	01	10	11
00	0	0	0	0.09
01	0	0	0	0.21
10	0	0	0	0.21
11	0	0	0	0.48

Table 5.12. Joint Probability Switching Estimate for Nodes 4899 and 4925 of C3540 for CR 40.

Node 4899	Node 4925			
State	00	01	10	11
00	0	0	0	0.09
01	0	0	0	0.21
10	0	0	0	0.21
11	0	0	0	0.48

CHAPTER 6

CONCLUSION

Here we are able to model a framework for learning input models and use them to estimate power and crosstalk. Though lot of research work has been done in both simulative nad probabilistic methods, they remain as two seperate entities. This work serves as a bridge effectively connecting the two methods. The technique proposed is unique and effectively models the spatial correlation and first order temporal correlation between the inputs. Also using Bayesian networks we are also able to include the causality between the nodes which cannot be modelled using undirected grpahs like Markov networks. Among the major advantages of this method are the excellent scalability to large circuits and the ability to model input dependencies using the concept of tree-dependent Bayesian Networks.

1. We present a unified graph based probabilistic framework for power estimation that takes into account high order spatial and first order temporal dependencies that are present in the inputs.

2. Modeling temporal correlation at the input nodes is automatic in way we set up the states of the random variables of the BN. For spatial correlation among the input nodes, we learn an approximate tree-dependent distributions in the primary inputs based on the input statistics.

3. The ease with which the Bayesian network is built considering the time taken makes the approach highly favorable over building Markov networks, as constructing the latter is both memory and time intensive.

4. We also tried other sampling methods using Matlab from Bayes Net Toolbox for matlab(BNT), an open source software by Kevin Murphy [54] and also the open source Probabilistic Network Library(PNL) from Intel [55], which is a C++ version of the Bayes Net toolbox for matlab by Kevin Murphy. But neither of them suited our purpose and so Hugin [23] was used.

5. Sampling the Bayesian network can also be done using Genie with any one of the approximate inference methods. The results obtained were similar and it can be used to sample a large Bayesian network.

6. Experimental results show that the method proposed by this thesis is highly efficient and effective in various ways namely the very less time taken to construct the input model and also sampling it, not to forget the very low error percentage and high compaction ratios.

Our future direction would be integarting this method with a vectorless approach. We intend to use this unified framework also with sequential circuits. Also, the present work models only first order temporal correlation. Dynamic Bayesian networks must be used to acheive higer order temporal correlation.

REFERENCES

[1] S. Bhanja, N. Ranganathan, "Switching Activity Estimation of VLSI circuits using Bayesian Networks", *IEEE Trans. on VLSI Systems*, 2003.

[2] C. Tsui, R. Marculescu, D. Marculescu, M. Pedram, "Improving the efficiency of power simulators by input vector compaction", *33rd Design Automation Conference*, pp. 165-168, June 1996.

[3] D. Marculescu, R. Marculescu, M. Pedram, "Stochastic Sequential Machines Synthesis with Application to Constrained Sequence Genereation ", *Proc. ACM/IEEE Design Automation Conference*, pp.696-701, June 1996.

[4] D. Marculescu, R. Marculescu, M. Pedram, "Stochastic Sequential Machines Synthesis with Application to Constrained Sequence Genereation ", *ACM Trans. on Design Automation of Electronic Systems*, Vol.5,No.2, Jan 2000.

[5] R. Marculescu, D. Marculescu, M. Pedram, "Vector Compaction using Dynamic Markov Models", *Technical Report CENG 96-14, Univ. of Southern California*, Feb. 1996

[6] R. Marculescu, D. Marculescu, M. Pedram, "Hierarchical Sequence Compaction for Power Estimation ", *Proc. ACM/IEEE Design Automation Conference*, Jun. 1997.

[7] R. Marculescu, D. Marculescu, M. Pedram, "Adaptive Models for Input Data Compaction for Power Simulators ",*Proc. ACM Asia and South-Pacific Design Automation Conference*, Japan, Jan. 1997.

[8] R. Radjassamy, J.D. Carothers "Vector Compaction for Efficient Simulation-Based Power Estimation", *IEEE Symposium on IC/Package Design Integration*, pp.138-142, Feb 1998.

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

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

[11] R. E. Neapolitan, "Learning Bayesian Networks", *Prentice Hall Series in Artificial Intelligence*, 2004.

[12] C. Hsu, W. Shen, "Vector Compaction for Power Estimation with Grouping and Consecutive Sampling Techniques", *IEEE International Symposium on Circiuts and Systems*, pp. 472-475 vol.2, May 2002.

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

[14] B. Ayari, B. Kaminska, “A New Dynamic Test Vector Compaction for Automatic Test Pattern Generation”, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp.353-358, March 1994.

[15] T.J. Lambert, K. Saluja, ”Methods for Dynamic Test Vector Compaction in Sequential Test Generation”, *Ninth International Conference on VLSI design*, pp.166-169, Jan.1996.

[16] J. Cheng, D. Bell, W. Liu, ” Learning Bayesian Networks from Data: An Efficient Approach Based on Information Theory”, *Technical Report, Department of Computer Science, University of Alberta*, 1998.

[17] J. Cheng, D. Bell. W. Liu, ”Learning Belief Networks from Data: An Information Theory Based Approach”, *Proceedings of the Sixth ACM International Conference on Information and Knowledge Management*, 1997.

[18] P. Spirtes, C. Glymour, R. Scheines, ”Causation, Prediction and Search”, 2nd Edition, *MIT Press*, 2000.

[19] S. Acid, L. M. Campos, ”An algorithm for finding minimum d-seperating sets in belief networks”, *Proceedings of the twelfth Conference of Uncertainty in Artificial Intelligence*, 1996.

[20] R. Burch, F. N. Najm, and T. Trick, “A Monte Carlo Approach for Power Estimation”, *IEEE Transactions on VLSI Systems*, vol.1-1, pp.63-71, March 1993.

[21] R. Marculescu, D. Marculescu, 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.

[22] D.Heckerman, D. Geiger, and D. Chickering, “Learning bayesian networks: The combination of knowledge and statistical data,” *Tech. Rep. MSR-TR-94-09, Microsoft Research*, 1994.

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

[24] C. Ababei, R. Marculescu, V. Sundarajan, “Probabilistic Aspects of Crosstalk Problems in CMOS ICs”, *Proc. IEEE Custom Integrated Circuits Conf.*, May 2000.

[25] K. T. Tang, E. G. Friedman, “Estimation of On-Chip Simultaneous Switching Noise on Signal Delay in Synchronous CMOS ICs”, *to be found.*. .

[26] V. Tiwari, D. Singh, S. Rajagopal, G. Mehta, R. Patel, F. Baez, “Reducing Power in High-performance Microprocessors”, *Proc. ACM/IEEE Design Automation Conference*, pp. 732-737, 1998.

[27] L. Benini, M. Favalli, P. Olivo, B. Ricco, “A Novel Approach to Cost-effective Estimate of Power Dissipation in CMOS ICs”, *European Design Automation Conference*, pp. 354-360, 1993.

[28] S. M. Kang, “Accurate Simulation of Power Dissipation in VLSI Circuits”, *Proc. IEEE Journal of Solid State Circuits*, vol. SC-21, no. 5, pp. 889-891, 1986.

[29] A. Murugavel, N. Ranganathan, R. Chandramouli, and S. Chavali, ”Average Power in Digital CMOS Circuits Using Least Square Estimation”, *Proc. of Intl. Conf. on VLSI Design*, Jan 2001.

[30] G. Y. Yacoub, and W. H. Ku, “An accurate simulation technique for shortcircuit power dissipation based on current component isolation,” *IEEE International Symposium on Circuits and Systems*, pp. 1157–1161, 1989.

[31] T. H. Krodel, “PowerPlay fast dynamic power estimation based on logic simulation,” *IEEE International Conference on Computer Design*, pp. 96–100, October 1991.

[32] A. C. Deng, Y. C. Shiau, and K. H. Loh, “Time Domain Current Waveform Simulation of CMOS Circuits”, *IEEE International Conference on Computer Aided Design*, Santa Clara, CA, pp. 208–211, Nov. 7-10, 1988.

[33] R. Tjarnstrom, “Power dissipation estimate by switch level simulation,” *IEEE International Symposium on Circuits and Systems*, pp. 881–884, May 1989.

[34] R. Burch, F. N. Najm, and T. Trick, “A Monte Carlo Approach for Power Estimation”, *IEEE Transactions on VLSI Systems*, vol. 1, no. 1, pp. 63–71, March 1993.

[35] F. N. Najm, and M. G. Xakellis, ”Statistical estimation of the switching activity in VLSI circuits,” *VLSI Design*, vol. 7, no. 3, pp. 243-254, 1998.

[36] K. Parker, and E. J. McCluskey, “Probabilistic Treatment of General Combinational Networks”, *IEEE Trans. on Computers*, vol. C, no. 24, pp. 668–670, June 1975.

[37] M. A. Cirit, “Estimating Dynamic Power Consumption of CMOS Circuits”, *IEEE International Conference on Computer -Aided Design*, pp. 534–537, 1987.

[38] R. E. Bryant, “Symbolic Boolean Manipulation with Ordered Binary-Decision Diagrams”, *ACM Computing Surveys*, vol. 24, no. 3, pp. 293–318, Sept. 1992.

[39] 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.

[40] S. Chakravarti, “ On the Complexity of using BDDs for the Synthesis and Analysis of Boolean Circuits”, *Proceedings of 27th Annual Conference on Communication, Control and Computing*, pp. 730–739, 1989.

[41] R. Burch, F. N. Najm, P. Yang, and D. Hocevar,”Pattern Independent Current Estimation for Reliability Analysis of CMOS Circuits”, *Proceedings of the 25th Design Automation Conference*, pp. 294–299, June, 1988.

[42] 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, no. 4, pp. 439–450, April 1990.

[43] 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.

[44] G. I. Stamoulis, and I.N. Hajj, “ Improved Techniques for Probabilistic Simulation including Signal Correlation Effects”, *Proceedings of the 30th ACM/IEEE Design Automation Conference*, pp. 379–383, June, 1993.

[45] 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, no. 2, pp. 310–323, February 1993.

[46] B. Kapoor, “Improving the Accuracy of Circuit Activity Measurement”, *Proc. ACM/IEEE Design Automation Conference*, pp. 734–739, June 1994.

[47] 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.

[48] 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.

[49] P. H. Schneider, U. Schlichtmann, and B. Wurth, “Fast power estimation of large circuits“, *IEEE Design & Test of Computers*, vol. 13, no. 1, pp. 70–78, Spring 1996.

[50] 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, no. 11, pp. 1099–1107, November 1998.

[51] T. Stohr, M. Alt, A. Hetzel, L. Koehl, “Analysis, Reduction and Avoidance of Crosstalk on VLSI Chips”, *International Symposium on Physical Design*, 1998.

[52] K. L. Shepard, V. Narayanan, R. Rose, “Harmony:Static Noise Analysis of Deep Submicron Digital ICs”, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 18, No. 08, Aug. 1999.

[53] Finn V. Jensen, “Bayesian Networks and Descision Graphs”, *Springer-Verlag New York, Inc.*, 2001.

[54] K. Murphy, “Bayes Net Toolbox for Matlab”, *Computing Science and Stastistics*, vol. 33 2001.

[55] `URL http://www.intel.com/research/mrl/pnl`

[56] M. Henrion, “Propagation of uncertainty by probabilistic logic sampling in Bayes’ networks,”*Uncertainty in Artificial Intelligence*, 1988.

[57] A. Macii, E. Macii, M. Poncino, R. Scarsi ”Stream Synthesis for Efficient Power Simulation Based on Spectral Transforms ”, *Proceedings of the International Symposium of Low Power Electronics and Design*, pp.30-35, 1998.