Efficient Estimation of Dynamic Power Dissipation Considering Glitches and its Application on Technology Mapping

Chi-Ying Tsui, Massoud Pedram,
and Alvin M. Despain

CENG Technical Report 93-12

Department of Electrical Engineering - Systems
University of Southern California
Los Angeles, California 90089-2562
(213)740-4458

Efficient Estimation of Dynamic Power Dissipation Considering Glitches and its Application on Technology Mapping

Chi-Ying Tsui, Massoud Pedram, Alvin M. Despain

12 April 1993

Abstract

In CMOS circuits, glitches contribute to the total power dissipation in a significant way. A low power synthesis tool should therefore account for this source of power dissipation. Existing power analysis techniques (based on symbolic simulation) are however very costly in terms of computation time and memory requirement. In this paper we present a fast and memory efficient power analysis technique for CMOS circuits which estimates the power dissipated due to glitches. We describe a linear time procedure for propagating the transition probabilities from circuit inputs to circuit outputs. Introducing the notion of transition waveforms, we prove that, under the assumption of uncorrelated inputs for internal gates in the circuit, our method is correct and exact. We then extend our procedure to account for input correlations by proposing a novel tagging mechanism based on the notion of steady state conditions for transition waveforms. Our method shows an average of 80% reduction in run-time over the exact method using symbolic simulation while the average error introduced is only 1%. We have incorporated this method in a technology mapper which minimizes the dynamic power dissipation under a real delay model. Preliminary results are encouraging.

Contents

1 Introduction **3**

2 Background and Terminology **5**

2.1 Power Dissipation Model 5

2.2 Power Dissipation and Transition Probabilities 6

2.3 Notation 7

2.3.1 Steady State Condition of a Transition Waveform 8

3 Propagation of Transition Probabilities **9**

3.1 Propagation through a Gate 9

3.1.1 Two-Input AND Gates with Uncorrelated Inputs 9

3.1.2 Two-input AND Gates with Correlated Inputs 11

3.2 Propagation through the Circuit 14

3.3 Power Estimation using Transition Probabilities 14

4 Technology Mapping using a Real Delay Model **15**

4.1 Tree Matching for Low Power 15

4.2 Discussion 16

4.3 Library Model Considering Charge Leakage 17

5 Experimental Results **18**

6 Concluding Remarks **19**

1 Introduction

With recent advances in microelectronic technology, smaller devices are now possible allowing more functionality on an integrated circuit (IC). Portable applications have shifted from conventional low performance products such as wristwatches and calculators to high throughput and computationally intensive products such as notebook computers and cellular phones. The new applications require high speed, yet low power consumption. In addition, power dissipation in chips rises with increased integration density and circuit speed. Low power design is thus essential to lower the packaging and cooling costs and prolong the life of the ICs. To achieve this goal, designers are willing to trade off area and performance for low power dissipation.

Shen et al. [8] present algorithms for reducing power dissipation during technology independent phase of logic synthesis. Intermediate nodes of an optimized multi-level implementation of a Boolean network are simplified so as to reduce the switching probabilities. Logic transformations which realize each node in the network as a disjoint cover are applied to further reduce the average power dissipation. Nodes that are not on the critical paths are partially collapsed and re-implemented in two-levels of logic. The rationale is that in general two-level sub-networks dissipate less power than their multi-level counterparts.

Prasad et al. [7] tackle the low power kernelization problem in multi-level logic minimization. During the factorization process, common sub-expressions which result in maximum reduction in switching activities are extracted. The result is a technology independent logic network with minimal total switching activities.

Tsui et al. [10] address the problem of minimizing the average power dissipation during the technology dependent phase of logic synthesis. The minimization consists of two steps. First a greedy algorithm which resembles the Huffman's algorithm is used to generate a NAND decomposition of an optimized Boolean network which minimizes the sum of average switching rates for all nodes in the network. Then a power efficient technology mapping algorithm is used to find an optimal power-delay trade-off value for a given timing constraints. Tiwari et al. [9] have also proposed a technology mapping paradigm for minimizing power and reported good results.

All the above low power logic synthesis algorithms are based on a power dissipation model which assumes a zero delay timing model. Under this model, the gates' outputs are switching at the same time and independently of the actual delay of the gates. Thus the output of each gate will switch at most once in every cycle according to its logic function. This assumption is too simplistic to capture the real world situations. Due to the delay of paths leading to the fanins of a gate, the inputs may arrive and switch at different times. The difference between the input switching times may cause hazards at the gate output and produce power consuming glitches. As an example, in Figure 1,

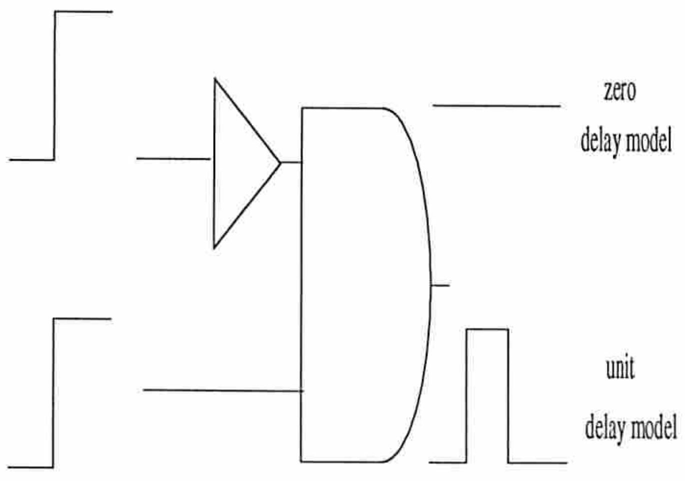


Figure 1: zero delay model vs general delay model

assuming a unit delay model, the output will switch twice instead of staying at zero as predicted by a zero delay model.

The zero delay model underestimates the actual average power dissipation of the circuit since it ignores the glitches. For some circuits, glitches contribute 20 - 70% of the total power dissipation in the circuit. Low power synthesis algorithm using a zero delay model can be therefore inaccurate in predicting the power dissipation cost of a particular optimization step and could pick up a sub-optimal solution. Thus, there is the need to use a more realistic model (using a general delay model) for estimating power dissipation during the synthesis.

Several researchers have studied the problem of estimating power dissipation under a general delay model. Najm el al. [6] use a probabilistic simulation approach to estimate the average current drawn by a circuit. Their approach is built on the notion of transition probabilities over a time period. They define the transition probability of a signal N at time t as the probability of a low-to-high (or high-to-low) transition from t^-(just before t) to t^+(just after t). Given transition probabilities (over time) at the circuit inputs, the transition probabilities (over time) at internal nodes are derived by propagating transition probabilities through nodes using an approach often employed by timing simulators. The main shortcoming of this approach is that during the propagation step, inputs to the gates (nodes) are assumed to be independent. This assumption does not hold for most of the internal nodes due to the presence of reconvergent fanout paths in general circuits.

Ghosh et al. [3] uses symbolic simulation in order to produce a set of Boolean functions which represent conditions for switching at each gate in the circuit at a specific time instance. Given input switching rates, the transition probability at each gate is calculated by performing a linear traversal of the Binary Decision Diagrams (BDDs) representation of the Boolean function constructed for the given time instance [5]. A general delay model which correctly computes the Boolean conditions that cause glitching is used and correlations due to the reconvergence of input signals are taken into account. This procedure is exact, however, suffers from excessive computation time and storage space. It can therefore be used as a final power analysis tool, but not in the inner loop of a synthesis program.

In order for a power estimation tool to be useful inside a synthesis environment, the following criteria have to be satisfied. First, it should be accurate enough, that is, it should account for glitches. Second, it should be fast since it is called many times during the synthesis process.

In this paper, we present a fast power estimation method using the notion of transition probabilities introduced in [5]. We describe a linear time algorithm which propagates the transition probabilities from the circuit inputs to the circuit outputs. For the case in which inputs to the gates are uncorrelated, our method is similar to [5]. We introduce the notion of transition waveforms to prove that this propagation mechanism is correct and exact. We then extend the method to account for input correlations by proposing a novel tagging mechanism which uses the notion of steady state conditions of the transition waveforms. This extension, although only approximately captures the input correlations during the transition probabilities propagation step, is very accurate (as confirmed by empirical data) and is valid under the most general delay models. This technique is then used during technology mapping for low power.

The rest of this paper is organized as follows. In Section 2 we review the notions of signal and transition probabilities and describe a model for power dissipation. Propagation of transition probabilities through the network is discussed in Section 3. In particular, we describe a novel tagging mechanism based on the steady state conditions of the input waveforms that allow us to capture the input correlations. A low power technology mapping algorithm which uses a real delay model for power estimation is detailed in Section 4. Experimental results and conclusions are presented in Sections 5 and 6 respectively.

2 Background and Terminology

2.1 Power Dissipation Model

The average power consumption for a single gate in a synchronous CMOS circuit is given by

$$P_{avg} = 0.5 \times C_{load} \times \frac{V_{dd}^2}{T_{cycle}} \times E(switching) \qquad (1)$$

where C_{load} is the load capacitance of the gate, V_{dd} is the supply voltage, T_{cycle} is the clock cycle time, and $E(switching)$ is the average number of switchings per cycle at the output of the gate.

$E(switching)$ is a function of the design style, the logic function being computed, the delay model, and switching probabilities of the primary inputs. Under a zero delay model, $E(switching)$ is always less than 1 while under a general delay model, $E(switching)$ can be larger than 1 due to hazards. For domino logic designs, switching probability of a node is equal to the probability of being 1 (for *p-type* circuit) or 0 (for *n-type* circuit). Since hazards do not occur for dynamic logic, $E(switching)$ is the same for general delay model and zero delay model . For static designs, $E(switching)$ is equal to the expected number of *0->1* and *1->0* transitions over one clock cycle.

2.2 Power Dissipation and Transition Probabilities

For average power estimation, we are interested in finding $E(switching)$ for a node. Given the transition probability of n at time t ($tp_n(t)$), $E(switching)$ is given by

$$E(switching) = \sum_{t=0}^{\infty} tp_n(t). \qquad (2)$$

Given transition probabilities at the primary inputs of a circuit, our technique propagates these probabilities through the network. Correlations among inputs to internal nodes are considered during the propagation.

Let us examine the operation of a circuit over the period of applying a sequence of two input vectors V_{-1}, V_0. Let V_{-1} be the vector applied at time $-\infty$ and V_0 be the vector applied at time 0. Clearly, when vector V_0 is applied, all circuit nodes have stablized to the values under V_{-1} (This timing model is referred to as the single stepping transition mode in [2]). During the time period between time 0 and time ∞, the logic value of an internal node n of a circuit varies over time according to the paths of the signal propagation from the primary inputs to n. We define the waveform showing the change in logic values of n over time from the initial state to the final steady state as *transition waveform* of n.

Consider a set Ω, each element of which represents a combination of logical waveforms to be applied to the circuit inputs. Elements of Ω are assigned occurrence probabilities. At the primary inputs there are four logical waveforms corresponding to *0->0, 0->1,1->0,* and *1->1.* For a circuit with P inputs, Ω has 4^P elements and therefore each intermediate node has 4^P transition waveforms. The transition probability $tp_n(t)$ is hence equal to the sum of the occurrence probabilities of transition waveforms at n which have *0->1* or *1->0* transitions at time t.

2.3 Notation

In this section, we will reiterate terminology and notation for transition waveforms. Similar definition can also be found in [6].

The following notation is used throughout the paper.

w_i	: a transition waveform i,
W_n	: the set of all w_is at node n,
$P_o(w_i)$	: the occurrence probability of w_i,
$sv_{w_i}(t)$	: the signal value of the transition waveform w_i at time t,
$up_{w_i}(t)$	: a binary value evaluating to 1 if w_i has a *0->1* transition at time t,
$down_{w_i}(t)$	: a binary value evaluating to 1 if w_i has a *1->0* transition at time t,
$sp_n(t)$	: signal probability of n at time t, i.e., the proability of n assuming a logic value 1 at time t,
$tpu_n(t)(tpd_n(t))$	: transition probability of n at time t, i.e., the probability of n having a *0->1* (or *1->0*) transition at time t.

The signal and transition probabilities are related by the following equations:

$$sp_n(t^+) = sp_n(t^-) + tpu_n(t) - tpd_n(t), \tag{3}$$

$$sp_n(t) = \sum_{w_i \in W_n} P_o(w_i) sv_{w_i}(t), \tag{4}$$

$$tpu_n(t) = \sum_{w_i \in W_n} P_o(w_i) up_{w_i}(t), \tag{5}$$

$$tpd_n(t) = \sum_{w_i \in W_n} P_o(w_i) down_{w_i}(t). \tag{6}$$

An example of some transition waveforms and some of the definitions (at time instance t) are shown in Figure 2.

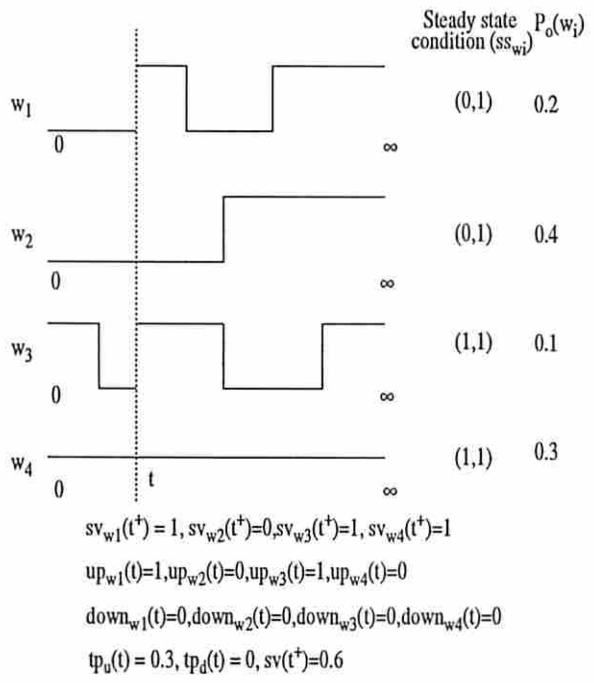

Steady state condition (ss_{wi})
$P_o(w_i)$
w_1 0 ∞ (0,1) 0.2
w_2 0 ∞ (0,1) 0.4
w_3 0 ∞ (1,1) 0.1
w_4 0 t ∞ (1,1) 0.3
$sv_{w1}(t^+) = 1, sv_{w2}(t^+)=0, sv_{w3}(t^+)=1, sv_{w4}(t^+)=1$
$up_{w1}(t)=1, up_{w2}(t)=0, up_{w3}(t)=1, up_{w4}(t)=0$
$down_{w1}(t)=0, down_{w2}(t)=0, down_{w3}(t)=0, down_{w4}(t)=0$
$tp_u(t) = 0.3, tp_d(t) = 0, sv(t^+)=0.6$

Figure 2: Transition waveforms

2.3.1 Steady State Condition of a Transition Waveform

We introduce the concept of steady state condition of a transition waveform next. A transition waveform w_i for node n corresponds to a specific primary input vector pair: V_{-1} and V_0. The steady states of w_i are defined as values at $t = 0^-$ (corresponding to steady state value of n under V_{-1}) and $t = \infty$ (corresponding to steady state value of n under V_0). These states are either (0,0), (0,1), (1,0) or (1,1), (see Figure (2)).
Let $x, y \in \{0, 1\}$, then

ss_{w_i} : steady state condition of w_i,
$W_n^{(x,y)}$: the set of all w_is at n such that $ss_{w_i} = (x, y)$,
and
W_n $= W_n^{(0,0)} + W_n^{(0,1)} + W_n^{(1,0)} + W_n^{(1,1)}$.

If primary inputs are temporally independent, i.e., the value at input i at clock cycle $k+1$ is independent of its value at clock cycle k, then the sum of the occurrence probabilities of all transition waveforms which have the same steady state condition (x, y) is equal to:

$$\sum_{w_i \in W_n^{(x,y)}} P_o(w_i) = P(x)P(y) \tag{7}$$

where $P(x)$ and $P(y)$ denote the probabilities of n assuming binary values x and y, respectively.

3 Propagation of Transition Probabilities

3.1 Propagation through a Gate

In this section, we describe how the transition probability at the output of a gate is derived given the signal and transition probabilities at its inputs. We prove the correctness and exactness of this method for uncorrelated inputs. In the case of correlated inputs, we give an approximate method which captures the input correlations by using steady state conditions of the transition waveforms for inputs. We use a two-input AND gate as an example. Similar results can be obtained for a two-input OR gate.[1]

3.1.1 Two-Input AND Gates with Uncorrelated Inputs

Let i_1, i_2 be the inputs and n be the output of a 2-input AND gate. Let w_{i_1} ,w_{i_2} and w_n be the transition waveforms at i_1, i_2 and n, respectively. w_n will have a *0->1* transition at time t if either w_{i_1} or w_{i_2} has a *0->1* transition and the other has a signal value equal to 1 or both have *0->1* transitions at time t. Hence we can write:

$$up_{w_n}(t) = up_{w_{i_1}}(t)sv_{w_{i_2}}(t^+) + up_{w_{i_2}}(t)sv_{w_{i_1}}(t^+) - up_{w_{i_1}}(t)up_{w_{i_2}}(t). \tag{8}$$

Since i_1 and i_2 are uncorrelated, so are w_{i_1} and w_{i_2}. Thus,

$$P_o(w_n) = P_o(w_{i_1} \wedge w_{i_2}) = P_o(w_{i_1})P_o(w_{i_2}). \tag{9}$$

Combining (5), (8) and (9) we can write

$$\begin{aligned} tpu_n(t) &= \sum_{w_{i_1} \in W_{i_1}} \sum_{w_{i_2} \in W_{i_2}} P_o(w_{i_1})P_o(w_{i_2}) \\ &\quad \left\{ up_{w_{i_1}}(t)sv_{w_{i_2}}(t^+) + up_{w_{i_2}}(t)sv_{w_{i_1}}(t^+) - up_{w_{i_1}}(t)up_{w_{i_2}}(t) \right\}, \end{aligned} \tag{10}$$

[1]Simple gates with $k > 2$ inputs are first decomposed into a tree of two-input gates and then processed. Complex gates are first decomposed into a two-level AND-OR representation with inverters. Propagation of transition propagation through an inverter is trivial.

Similarly,

$$tpd_n(t) = \sum_{w_{i_1} \in W_{i_1}} \sum_{w_{i_2} \in W_{i_2}} P_o(w_{i_1}) P_o(w_{i_2})$$
$$\left\{ down_{w_{i_1}}(t) sv_{w_{i_2}}(t^-) + down_{w_{i_2}}(t) sv_{w_{i_1}}(t^-) - down_{w_{i_1}}(t) down_{w_{i_2}}(t) \right\}. \quad (11)$$

Theorem 3.1

Given uncorrelated inputs i_1 and i_2, transition probabilities at the output of a two-input AND gate n are given by

$$tpu_n(t) = tpu_{i_1}(t) sp_{i_2}(t^+) + tpu_{i_2}(t) sp_{i_1}(t^+) - tpu_{i_1}(t) tpu_{i_2}(t) \quad (12)$$

and

$$tpd_n(t) = tpd_{i_1}(t) sp_{i_2}(t^-) + tpd_{i_2}(t) sp_{i_1}(t^-) - tpd_{i_1}(t) tpd_{i_2}(t) \quad (13)$$

Proof:

Using (4) and (5), we can write (10) as

$$\begin{aligned} tpu_n(t) &= \sum_{w_{i_1} \in W_{i_1}} P_o(w_{i_1}) up_{w_{i_1}}(t) \sum_{w_{i_2} \in W_{i_2}} P_o(w_{i_2}) sv_{w_{i_2}}(t^+) + \\ &\quad \sum_{w_{i_1} \in W_{i_1}} P_o(w_{i_1}) sv_{w_{i_1}}(t^+) \sum_{w_{i_2} \in W_{i_2}} P_o(w_{i_2}) up_{w_{i_2}}(t) - \\ &\quad \sum_{w_{i_1} \in W_{i_1}} P_o(w_{i_1}) up_{w_{i_1}}(t) \sum_{w_{i_2} \in W_{i_2}} P_o(w_{i_2}) up_{w_{i_2}}(t) \\ &= tpu_{i_1}(t) sp_{i_2}(t^+) + tpu_{i_2}(t) sp_{i_1}(t^+) - tpu_{i_1}(t) tpu_{i_2}(t) \end{aligned}$$

The derivation of $tpd_n(t)$ is similar.
□

Equations (12 - 13) allow us to obtain closed form expressions for $tpu_n(t)$ and $tpd_n(t)$. In the next section, we derive approximate closed form expressions for nodes with correlated inputs using the notion of steady state conditions.

3.1.2 Two-input AND Gates with Correlated Inputs

If the input signals are correlated, their corresponding transition waveforms are also correlated. Equation (9) has to be replaced by:

$$P_o(w_{i_1} \wedge w_{i_2}) = P_o(w_{i_1})P_o(w_{i_2}|w_{i_1}) \tag{14}$$

where $P_o(w_{i_2}|w_{i_1})$ is the conditional probability of w_{i_2} given w_{i_1}. The correlation between two transition waveforms at two nodes depends on the logical waveforms applied to the primary inputs and the circuit topology. It is expensive to compute the exact correlation. We instead use the correlation between the steady state conditions of transition waveforms to *approximate* the actual correlation as detailed next

<u>Correlation between Transition Waveforms</u>
We approximate the correlation between two transition waveforms (w_{i_1},w_{i_2}) by assuming two waveforms are correlated only by their steady state conditions ($ss_{w_{1_1}}$,$ss_{w_{i_2}}$). Then

$$\frac{P_o(w_{i_1} \wedge w_{i_2})}{P_o(w_{i_1})P_o(w_{i_2})} = \frac{P_o(ss_{w_{i_1}} \wedge ss_{w_{i_2}})}{P_o(ss_{w_{i_1}})P_o(ss_{w_{i_2}})}, \tag{15}$$

and

$$P_o(w_{i_1} \wedge w_{i_2}) = \frac{P_o(ss_{w_{i_1}} \wedge ss_{w_{i_2}})}{P_o(ss_{w_{i_1}})P_o(ss_{w_{i_2}})} P_o(w_{i_1})P_o(w_{i_2}). \tag{16}$$

Let $ss_{w_{i_1}} = (x_1, y_1)$ and $ss_{w_{i_2}} = (x_2, y_2)$. If primary inputs are independent, then

$$P_o(ss_{w_{i_1}}) = P(i_1 = x_1)P(i_1 = y_1), \tag{17}$$

and

$$P_o(ss_{w_{i_1}} \wedge ss_{w_{i_2}}) = P(i_1 = x_1 \wedge i_2 = x_2)P(i_1 = y_1 \wedge i_2 = y_2). \tag{18}$$

$P(i_1 \wedge i_2)$ can be calculated by building a BDD for the function $i_1 \wedge i_2$ and doing a depth first traversal on the BDD to find the signal probability of the function [5].

<u>Transition Probabilities and Steady State Conditions</u>
Each transition waveform is tagged by its steady states (x, y). Similarly, we can tag the signal and transition probabilities. From (4), we have

$$\begin{aligned} sp_n(t) &= \sum_{(x,y)\in\{(0,0),(0,1),(1,0),(1,1)\}} \sum_{w_i \in W_n^{(x,y)}} P_o(w_i) sv_{w_i}(t) \\ &= sp_n^{(0,0)}(t) + sp_n^{(0,1)}(t) + sp_n^{(1,0)}(t) + sp_n^{(1,1)}(t), \end{aligned} \tag{19}$$

where

$$sp_n^{(x,y)}(t) = \sum_{w_i \in W_n^{(x,y)}} P_o(w_i) sv_{w_i}(t). \tag{20}$$

OUTPUT TAGS	INPUT TAGS
(0,0)	[(0,0),(0,0)], [(0,0),(0,1)], [(0,0),(1,0)],[(0,0),(1,1)],[(0,1),(0,0)] [(0,1),(1,0)],[(1,0),(0,0)],[(1,0),(0,1)],[(1,1),(0,0)]
(0,1)	[(0,1),(0,1)], [(0,1),(1,1)],[(1,1),(0,1)]
(1,0)	[(1,0),(1,0)], [(1,0),(1,1)],[(1,1),(1,0)]

Table 1: The TAG table for a two-input AND gate

(5) and (6) then become:

$$tpu_n(t) = tpu_n^{(0,0)}(t) + tpu_n^{(0,1)}(t) + tpu_n^{(1,0)}(t) + tpu_n^{(1,1)}(t), \tag{21}$$

$$tpd_n(t) = tpd_n^{(0,0)}(t) + tpd_n^{(0,1)}(t) + tpd_n^{(1,0)}(t) + tpd_n^{(1,1)}(t), \tag{22}$$

where

$$tpu_n^{(x,y)}(t) = \sum_{w_i \in W_n^{(x,y)}} P_o(w_i) up_{w_i}(t), \tag{23}$$

$$tpd_n^{(x,y)}(t) = \sum_{w_i \in W_n^{(x,y)}} P_o(w_i) down_{w_i}(t). \tag{24}$$

We refer to $sp_n^{(x,y)}(t)$, $tpu_n^{(x,y)}(t)$ and $tpd_n^{(x,y)}(t)$ as the tagged signal and transition probabilities of n at time t with tag (x,y).

Propagation of Transition Proabilities

The tag of a waveform at the output of a two-input AND gate depends on tags of waveforms at its inputs. A waveform at the output with tag (1,1) requires both input waveforms to have (1,1) tags. Table 1 gives the input tag pair that will give rise to the specified output tag for a two-input AND gate. The set of input tag pairs that forces the output tag to be *tag* is denoted as the *forcing tag set* of *tag (FTS(tag))*.

Theorem 3.2

Given the $tpu_i^{tag_i}(t)$ and $sp_i^{tag_i}(t)$ at the inputs, the transition probabilities at the output of a two-input AND gate with a tag *tag* is given by

$$tpu_n^{tag}(t) = \sum\nolimits_{(tag_1,tag_2)\in FTS(tag)} \frac{P_o(ss_{w_{i_1}}=tag_1 \wedge ss_{w_{i_2}}=tag_2)}{P_o(ss_{w_{i_1}}=tag_1)P_o(ss_{w_{i_2}}=tag_2)} \left\{tpu_{i_1}^{tag_1}(t)sp_{i_2}^{tag_2}(t^+) + tpu_{i_2}^{tag_2}(t)sp_{i_1}^{tag_1}(t^+) - tpu_{i_1}^{tag_1}(t)tpu_{i_2}^{tag_2}(t)\right\} \tag{25}$$

and

$$tpd_n^{tag}(t) = \sum\nolimits_{(tag_1,tag_2)\in FTS(tag)} \frac{P_o(ss_{w_{i_1}}=tag_1 \wedge ss_{w_{i_2}}=tag_2)}{P_o(ss_{w_{i_1}}=tag_1)P_o(ss_{w_{i_2}}=tag_2)} \left\{tpd_{i_1}^{tag_1}(t)sp_{i_2}^{tag_2}(t^-) + tpd_{i_2}^{tag_2}(t)sp_{i_1}^{tag_1}(t^-) - tpd_{i_1}^{tag_1}(t)tpd_{i_2}^{tag_2}(t)\right\} \tag{26}$$

Proof:

We prove (25) for the case of tag = (1,1). Similar proofs can be given for other tags.

From (5) we have:

$$\begin{aligned}
tpu_n^{(1,1)}(t) &= \sum_{w_{i_1} \in W_{i_1}^{(1,1)}} \sum_{w_{i_2} \in W_{i_2}^{(1,1)}} P_o(w_{i_1} \wedge w_{i_2}). \\
&\quad \left\{ up_{w_{i_1}}(t) sv_{w_{i_2}}(t^+) + up_{w_{i_2}}(t) sv_{w_{i_1}}(t^+) - up_{w_{i_1}}(t) up_{w_{i_2}}(t) \right\} \\
&= \frac{P_o(ss_{w_{i_1}} = (1,1) \wedge ss_{w_{i_2}} = (1,1))}{P_o(ssw_{i_1} = (1,1)) P_o(ss_{w_{i_2}} = (1,1))}. \\
&\quad (\sum_{w_{i_1} \in W_{i_1}^{(1,1)}} P_o(w_{i_1}) up_{w_{i_1}}(t) \sum_{w_{i_2} \in W_{i_2}^{(1,1)}} P_o(w_{i_2}) sv_{w_{i_2}}(t^+) + \\
&\quad \sum_{w_{i_1} \in W_{i_1}^{(1,1)}} P_o(w_{i_1}) sv_{w_{i_1}}(t^+) \sum_{w_{i_2} \in W_{i_2}^{(1,1)}} P_o(w_{i_2}) up_{w_{i_2}}(t) - \\
&\quad \sum_{w_{i_1} \in W_{i_1}^{(1,1)}} P_o(w_{i_1}) up_{w_{i_1}}(t) \sum_{w_{i_2} \in W_{i_2}^{(1,1)}} P_o(w_{i_2}) up_{w_{i_2}}(t)) \\
&= \frac{P_o(ss_{w_{i_1}} = (1,1) \wedge ss_{w_{i_2}} = (1,1))}{P_o(ss_{w_{i_1}} = (1,1)) P_o(ss_{w_{i_2}} = (1,1))} \\
&\quad \left\{ tpu_{i_1}^{(1,1)}(t) sp_{i_2}^{(1,1)}(t^+) + tp_{i_2}^{(1,1)}(t) sp_{i_1}^{(1,1)}(t^+) - tp_{i_1}^{(1,1)}(t) tp_{i_2}^{(1,1)}(t) \right\}.
\end{aligned}$$

The proof for (26) is similar. □

Similar expressions can be derived for $tpu_n^{(0,1)}(t)$,$tpu_n^{(1,0)}(t)$,$tpu_n^{(0,0)}(t)$,$tpd_n^{(0,1)}(t)$,$tpd_n^{(1,0)}(t)$ and $tpd_n^{(0,0)}(t)$ using the tagged signal and transition probabilities at the inputs. For example,

$$\begin{aligned}
tpu_n^{(0,1)}(t) &= \frac{P_o(ss_{w_{i_1}} = (0,1) \wedge ss_{w_{i_2}} = (1,1))}{P_o(ss_{w_{i_1}} = (0,1)) P_o(ss_{w_{i_2}} = (1,1))}. \\
&\quad \left\{ tpu_{i_1}^{(0,1)}(t) sp_{i_2}^{(1,1)}(t^+) + tp_{i_2}^{(0,1)}(t) sp_{i_1}^{(1,1)}(t^+) - tp_{i_1}^{(0,1)}(t) tp_{i_2}^{(1,1)}(t) \right\} + \\
&\quad \frac{P_o(ss_{w_{i_1}} = (1,1) \wedge ss_{w_{i_2}} = (0,1))}{P_o(ss_{w_{i_1}} = (1,1)) P_o(ss_{w_{i_2}} = (0,1))}. \\
&\quad \left\{ tpu_{i_1}^{(1,1)}(t) sp_{i_2}^{(0,1)}(t^+) + tp_{i_2}^{(1,1)}(t) sp_{i_1}^{(0,1)}(t^+) - tp_{i_1}^{(1,1)}(t) tp_{i_2}^{(0,1)}(t) \right\} + \\
&\quad \frac{P_o(ss_{w_{i_1}} = (0,1) \wedge ss_{w_{i_2}} = (0,1))}{P_o(ss_{w_{i_1}} = (0,1)) P_o(ss_{w_{i_2}} = (0,1))}. \\
&\quad \left\{ tpu_{i_1}^{(0,1)}(t) sp_{i_2}^{(0,1)}(t^+) + tp_{i_2}^{(0,1)}(t) sp_{i_1}^{(0,1)}(t^+) - tp_{i_1}^{(0,1)}(t) tp_{i_2}^{(0,1)}(t) \right\}.
\end{aligned} \tag{27}$$

3.2 Propagation through the Circuit

Given the tagged signal and transition probabilities at the primary inputs, the tagged probabilities at the internal nodes can be derived using the following procedure. We visit nodes in the network in topological fashion (from primary inputs to primary outputs). When we reach a node n, the tagged signal and transition probabilities at its inputs are already known. k-input ($k > 2$) and complex gates are decomposed into inverters, two-input AND, and two-input OR gates before propagation. Note that this decomposition is done on a separate copy of the node in the network. Nodes in the networks are <u>not</u> actually decomposed during the analysis. After waveform propagation, the time axis for the output waveform is shifted forward by an amount equal to the delay of n.

For a general delay model, the gate's inertia is taken into account as follows. If the inertial delay of a gate is non-zero, then all inputs to the gate must remain constant for a period at least equal to the inertial delay. The transition of an input at time t will not be propagated to the output if there is an opposite transition within the period of the inertial delay from of the gate. This is an important consideration, since by ignoring the inertial delay of gates, we may overestimate the dynamic power dissipation in the circuit.

3.3 Power Estimation using Transition Probabilities

The average power dissipated by a CMOS gate n is given by

$$Power(n) = 0.5 \times C_{load} \times \frac{V_{dd}^2}{T_{cycle}} \times E_n(switching) \tag{28}$$

where

$$E_n(switching) = \sum_{t=0}^{T_{cycle}} tpu_n(t) + tpd_n(t). \tag{29}$$

Using the procedure described in Section 3.2, we calculate the transition probabilities at all nodes in the network and sum over to obtain the total power dissipation in the network. The assumption is, of course, that the clock cycle time is long enough for all transitions (including hazards) to settle.

In [3], the transition probabilities are calculated using symbolic simulation which uses the global function of a node (in terms of primary inputs) in order to estimate the transitions at the output of the node. Our method estimates the transition probabilities of a node by two closed form formula (25) and (26) which only **depend on the immediate inputs of the node**. This makes our algorithm very attractive for use during synthesis. As an example, we will describe a low power technology mapping algorithm which uses transition probabilities as calculated by our procedure.

4 Technology Mapping using a Real Delay Model

Given a Boolean network representing a combinational logic circuit optimized by technology independent synthesis procedures and a target library, we bind nodes in the network to gates in the library such that average power consumption of the final implementation is minimized and timing constraints are satisfied. The idea is to reduce technology mapping to DAG covering and to approximate DAG covering by a sequence of tree coverings which can be performed optimally using dynamic programming [4].

The low power technology mapper described here follows a procedure similar to [10], with the difference that a general delay model is used (instead of a zero delay model). A quick overview of the procedure is given next. First a postorder traversal is used to construct a power-delay trade-off curve. Next, a preorder traversal is performed to determine the mapping solution that minimizes the average power subject to the require time constraints. In the following sections we give a brief description of the implications of using a general delay model instead of a zero delay model. For the details of the mapping algorithm, refer to [10] and [1].

4.1 Tree Matching for Low Power

Consider a match g at node n of a NAND-decomposed tree. The inputs to node n consist of nodes n_i which fanout to node n. The nodes which are covered by match g are denoted by $merged(n, g)$. The nodes which are not in $merged(n, g)$ but fanin to $merged(n, g)$ are denoted by $inputs(n, g)$. The *mapped-parent*(n_i) is some node n for which there exists a matching gate g such that $n_i \in inputs(n, g)$.

With each node in the network we store a power-delay curve. A point on the curve represents the arrival time at the output of the node and the average power dissipated in its mapped transitive fanin cone. Points on the curve represent various mapping solutions with different tradeoffs between average power and speed. Each point is annotated with the corresponding transition probabilities. These probabilities are calculated using the procedure detailed in Section 3.

We have adopted the pin-dependent SIS library delay model for calculation of the arrival time and power dissipation as follows. Suppose that gate g has matched at node n, then the output arrival time at n is given by

$$arrival(n, g, C_n) = max_{n_i \in inputs(n,g)}(R_{i,g}(C_n + C_{n,g}^{int}) + arrival(n_i, g_i, C_i)) \qquad (30)$$

where $R_{i,g}$ is the *drive* resistance of g corresponding to a signal transition at input i, $C_{n,g}^{int}$ is the intrinsic load of matching gate g at node n,[2] C_n is the load capacitance seen

[2]Note that the intrinsic gate delay, $\tau_{i,g}$, is equal to $R_{i,g}C_{n,g}^{int}$.

at n, $arrival(n_i, g_i, C_i)$ is the arrival time at input i corresponding to load C_i seen at that input, and g_i is the best match found at input i.

In [10] the dynamic value of average power dissipation at node n is given by

$$power(n,g) = 0.5C_{n,g}^{int}\frac{V_{dd}^2}{T_{cycle}}E_n + \sum_{n_i \in inputs(n,g)} (0.5C_{n_i}^{load}\frac{Vdd^2}{T_{cycle}}E_{n_i} + power(n_i, g_i)) \quad (31)$$

where E_{n_i} is the average switching activity at the output of gate n_i, C_{n_i} is the output load capacitance seen at node n_i, T_{cycle} is the cycle time, V_{dd} is the power supply voltage, and $power(n_i, g_i)$ is the average power dissipated at input i. This is based on the observation that under a zero-delay model, E_n depends on the global function of n and not its particular implementation. Hence, E_n is independent of the gate matching at n. The average power contribution of n's output load will be included at *mapped-parent(n)*.

However, under a real-delay model, E_n depends on the type and delay of the gate. The transition probabilities at n have to be shifted forward according to the delay. Hence (31) is not applicable. Instead, we use

$$power(n,g) = 0.5(C_n + C_{n,g}^{int})\frac{Vdd^2}{T_{cycle}}E_n + \sum_{n_i \in inputs(n,g)} power(n_i, g_i) \quad (32)$$

When calculating the average power dissipation at node n, the gate driven by n is not yet mapped and hence C_n and the actual delay seen at n are not yet determined[3]. Hence a *default* load capacitance is assumed during the calculations of $Power(n, g)$, E_n,and the propagation of transition probabilities. Once *mapped-parent(n)* is mapped, C_n has to be recalculated (similar to the timing recalculation procedure described [10]) in order to account for the difference between the estimated and actual loads. After the actual C_n is determined, the arrival time seen at n is re-calculated. The transition waveform of n has to be shifted using the new delay of node n and E_n is recalculated. The updated transition probabilities are then used to calculate the transition probabilities at the *mapped-parent(n)*.

4.2 Discussion

Since the zero-delay model does not account for power dissipated due to glitches, it may pick a sub-optimal solution instead. This shortcoming is overcome by using a general delay model when calculating the expected number of transitions. However, there is a

[3]This is usually referred to as the "unknown load problem".

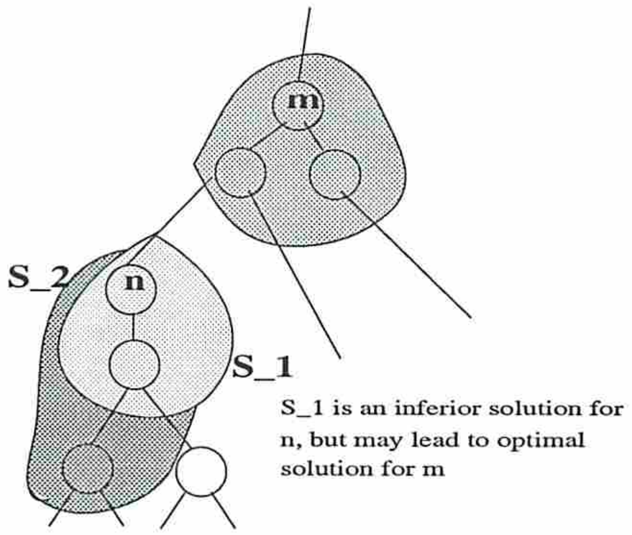
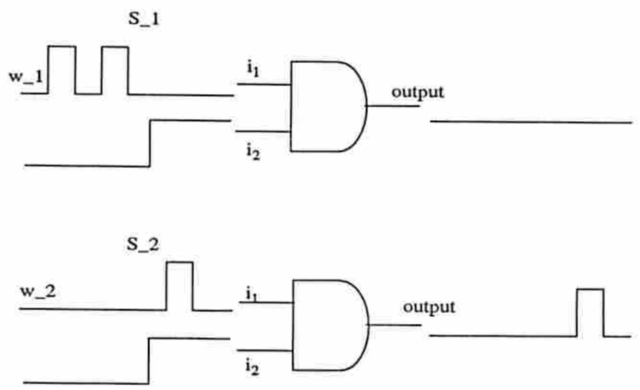

m
n
S_2
S_1
S_1 is an inferior solution for n, but may lead to optimal solution for m

Figure 3: Mapping Example

shortfall with using the general delay model during the dynamic programming based tree mapping. The mapping algorithm assumes that the current best solution is derived from the best solutions stored at the fanin nodes to the matching gate. This is true for power estimation under a zero delay model, but not for power estimation under a general delay model. Consider Figure (3) as an example. Let S_1 and S_2 be two mapping candidates at n with the same delay. Let n have a larger $E_n(switching)$ value for S_1. It appears that S_2 is superior to S_1 and thus S_1 is dropped form the power-delay curve. However, when doing mapping at m, the mapped-parent of n, due to the difference in the transition waveform timing for S_1 and S_2, the best solution at m may be coming from S_1 instead of S_2 (see Figure (4)), that is, if S_1 is dropped from the power-delay curve, the optimal solution may not be found. It is very expensive to store all partial solutions, so we still drop the locally inferior solutions knowing that dynamic programming approach will then only act as a heuristic approach.

4.3 Library Model Considering Charge Leakage

In most of the power dissipation models, it is assumed that power is dissipated only during the charging and discharging of the intrinsic and output load driven by the gate. This model does not consider the power dissipation due to charging of the source and drain capacitances of the internal nodes of a complex gate. The charge leakage phenomenon is illustrated by an example in Figure 5. For a two-input NOR gate, in the conventional power dissipation model, power is dissipated when the output is switching. So if the input is changing from 01 to 10, the output remains unchanged and so no power is dissipated. However, when the input is 01, the capacitance of node A



Let S_1 and S_2 be two partial solutions for i_1 and w_1 and w_2 be two transition waveforms for S_1 and S_2 respectively. Considering only w_1 and w_2, S_1 seems to be inferior to S_2 since it has more transitions at i_1 which means more power dissipation. But S_1 will give a better solution at the output. So it is not necessary inferior.

Figure 4: Example showing an optimal partial solution for a node may lead to a sub-optimal partial solution for the mapping of its mapped fanout

is charged and it is discharged when the input is changed to 10. This contributes to the power dissipation in the circuit. The amount of power dissipation depends on the source and drain capacitances and hence the sizes of the transistors. To account for this source of power dissipation during the technology mapping, more realistic models of the gates in library and power calculation procedures are required. We are in the process of developing such models.

5 Experimental Results

We compared our power estimation procedure with that of [3] in terms of accuracy and performance. We applied both algorithms on a subset of the ISCAS-89 and the MCNC-91 benchmark sets. All circuits were mapped using the SIS mapper. All primary inputs were assumed to be independent. 20MHz clock frequency is assumed and all power estimates are in micro-Watts. The delays and loads for circuits were obtained using the pin-dependent SIS library delay model for *lib2.genlib*.

Table 2 contains our experimental results showing the average power dissipation estimation and run time for both unit delay and real delay model. All run times (in seconds) were obtained on a Sun Sparc 1+ with 64Mb of memory. The entry N/A means that results could not be obtained due to excessive memory requirements. It is seen that the run times are lower than symbolic simulation by an average of 80% while the average error introduced by our approximation is only 1%. For large benchmarks, symbolic

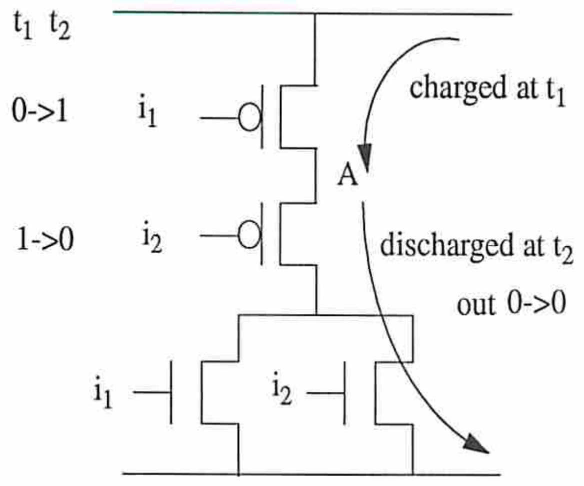


Figure 5: Example showing the effect of charge leakage on power dissipation

simulation cannot even complete the estimation.

To see the impact of using a real delay model versus a zero delay model during technology mapping, we applied our low power technology mapper [10] on a small subset of the ISCAS-89 and the MCNC-91 benchmarks using different timing models. The preliminary results show some improvement. We are currently working on the full implementation of this mapper (which should also account for the leakage power dissipation). Detailed results will be presented in the final paper.

6 Concluding Remarks

We presented an efficient algorithm for average power estimation using transition probability propagation. We proved that the algorithm is exact for uncorrelated inputs and gave an approximate procedure for correlated input using the notion of steady state conditions of transition waveform. Experimental results show that significant reduction in run time is achieved while maintaining a high degree of accuracy.

We also studied the effect of using real delay model in low power technology mapping. The need for a more realistic library and power dissipation model considering charge leakage in the internal nodes of a complex gate was discussed. Our future goal is to develop and incorporate these models in our low power synthesis and mapping environment.

cir-uit	UNIT DELAY						GENERAL DELAY					
	POWER ESTIMATION			RUN TIME			POWER ESTIMATION			RUN TIME		
	ss	tp	% err.	ss	tp	% red.	ss	tp	% err.	ss	tp	% red.
count	196.8	195.1	0.8	20.9	2.2	89.3	193.4	190.8	1.38	8.69	1.53	69.7
cordic	180.0	184.5	2.4	36.1	1.4	95.9	167.8	171.0	1.92	8.69	1.53	69.7
s208	133.6	132.9	0.5	5.7	1.1	80.7	129.9	129.0	0.72	4.0	1.2	69.7
s298	243.2	246.0	1.1	8.9	1.7	80.6	231.4	232.9	0.64	4.1	1.8	56.3
s349	265.3	263.4	0.7	18.8	2.0	89.1	231.0	224.3	2.92	11.1	2.2	79.8
s382	291.4	294.8	1.1	10.7	1.9	81.9	275.3	279.0	1.34	6.4	2.0	68.2
s386	324.2	322.3	0.5	11.0	2.4	78.0	311.8	309.3	0.80	6.0	2.5	57.6
s400	303.2	306.3	1.0	11.1	1.9	82.3	286.6	289.6	1.07	6.6	2.1	67.5
s444	329.2	326.5	0.8	15.5	2.4	84.4	312.7	308.7	1.26	8.7	2.7	68.8
s526	449.9	452.9	0.6	16.8	3.1	81.3	426.3	427.8	0.35	7.5	3.4	54.5
s641	323.0	322.3	0.2	100.2	3.9	96.1	299.1	288.0	3.69	31.8	4.6	85.2
s713	342.5	344.1	0.4	92.5	4.2	95.4	314.3	304.2	3.23	35.9	5.1	85.7
s820	764.4	771.3	0.9	29.5	6.0	79.4	737.5	739.8	0.32	17.3	6.5	62.5
s832	787.9	794.2	0.8	31.9	6.1	80.7	759.6	761.5	0.25	25.7	6.6	74.1
s953	475.5	473.3	0.4	85.9	6.3	92.5	434.5	428.3	1.43	33.8	7.0	79.0
s1196	853.8	844.3	1.1	258.3	13.4	94.8	762.8	754.2	1.13	110.4	15.5	85.9
s1238	928.7	921.1	0.8	640.4	12.7	98.0	832.2	822.8	1.13	145.4	14.7	89.8
s1488	1321.6	1322.1	0.0	87.4	11.1	87.3	1202.5	1201.8	0.06	44.3	12.3	72.2
s1494	1339.4	1337.9	0.1	80.0	11.3	85.8	1218.6	1215.2	0.28	418.8	12.6	96.9
s5378	N/A	2467.7	-	N/A	30.3	-	2198.0	2263.5	2.98	289.4	33.3	88.49
s838	N/A	476.7	-	N/A	6.8	-	N/A	464.6	-	N/A	7.6	-
C880	N/A	733.7	-	N/A	72.1	-	N/A	651.4	-	N/A	75.6	-
C1908	N/A	1643	-	N/A	202	-	N/A	1064	-	N/A	209	-
	AVE. %ERR.		0.78	AVE. %RED.		87.0	AVE. %ERR.		1.34	AVE. %RED.		73.8

Table 2: Comparison of the Power Estimation and Run-time: ss : using symbolic simulation, tp : using transition probability

References

[1] K. Chaudhary and M. Pedram. A near-optimal algorithm for technology mapping minimizing area under delay constraints. In *Proceedings of the 29th Design Automation Conference*, pages 492–498, June 1992.

[2] S. Devadas, K. Keutzer, S. Malik, and A. Wang. Certified timing verification and the transition delay of a logic circuit. In *Proceedings of the 29th Design Automation Conference*, pages 549–555, June 1992.

[3] A. A. Ghosh, S. Devadas, K. Keutzer, and J. White. Estimation of average switching activity in combinational and sequential circuits. In *Proceedings of the 29th Design Automation Conference*, pages 253–259, June 1992.

[4] K. Keutzer. DAGON: Technology mapping and local optimization. In *Proceedings of the Design Automation Conference*, pages 341–347, June 1987.

[5] F. Najm. Transition density, a stochastic measure of activity in digital circuits. In *Proceedings of the 28th Design Automation Conference*, pages 644–649, June 1991.

[6] F. N. Najm, R. Burch, P. Yang, and I. Hajj. Probabilistic simulation for reliability analysis of CMOS VLSI circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 9 No. 4:439–450, April 1990.

[7] S. C. Prasad and K. Roy. Circuit activity driven multilevel logic optimization for low power reliable operation. In *Proceedings of the European Design Automation Conference*, pages 368–372, February 1993.

[8] A. A. Shen, A. Ghosh, S. Devadas, and K. Keutzer. On average power dissipation and random pattern testability of CMOS combinational logic networks. In *Proceedings of the IEEE International Conference on Computer Aided Design*, November 1992.

[9] V. Tiwari, P. Ashar, and S. Malik. Technology mapping for low power. In *Proceedings of the 30th Design Automation Conference*, June 1993.

[10] C. Y. Tsui, M. Pedram, and A. Despain. Technology decomposition and mapping targeting low power dissipation. In *Proceedings of the 30th Design Automation Conference*, June 1993.