Power Efficient Technology Decomposition and Mapping under an Extended Power Consumption Model

Chi-Ying Tsui, Massoud Pedram, Alvin M. Despain
Department of Electrical Engineering - Systems
University of Southern California
Los Angeles, CA 90089

Manuscript received ___

The authors are with the Department of Electrical Engineering-System, University of Southern California, Los Angeles, CA 90089.

Contents

List of Figures

List of Tables

Abstract

We propose a new power consumption model which accounts for the power consumption at the internal nodes of a CMOS *gate. Next, we address the problem of minimizing the average power consumption during the technology dependent phase of logic synthesis. Our approach consists of two steps. In the first step, we generate a* NAND *decomposition of an optimized Boolean network such that the sum of average switching rates for all nodes in the network is minimum. In the second step, we perform a power efficient technology mapping that finds a minimal power mapping for given timing constraints (subject to the unknown load problem).*

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 as for such products longer battery life translates to extended use and better marketability. With the convergence of telecommunications, computers, consumer electronics, and biomedical technologies, the number of low power applications is expected to grow. Another driving force behind design for low power is that excessive power consumption is becoming the limiting factor in integrating more transistors on a single chip or on a multi-chip module due to cooling, packaging and reliability problems. Exploring the trade-offs between area, performance and power during synthesis and design is thus demanding more attention.

1.1 Prior Work

Many researchers (in solid state circuits and technology areas) have been studying low power/ low voltage design techniques. For example, research is being conducted in low power DRAM and SRAM designs, aggressive voltage scaling and process optimization for active logic circuits, device modeling and simulation tools for low power circuits, low power analog circuit design, etc. Other researchers (in computer architecture area) are exploring instruction set architectures and novel memory management schemes for low power, processor design using self-clocking, static and dynamic power management strategies, etc. The computer aided design community has recently started paying more attention to power estimation and low power design.

Several researchers have studied the problem of estimating power consumption. Cirit [6] estimates the dynamic power consumption at the transistor level based on the probability of the drain of a transistor making a power consuming transition. Burch et al. [4] introduce the concept of probability waveforms. Given such waveforms at the primary inputs and with some convenient partitioning of the circuit, they examine every sub-circuit and derive corresponding waveforms at the internal circuit nodes. Najm et al. [15] 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. 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. These

methods assume inputs to sub-circuits are independent and thus do not account for the reconvergent fanout and input correlations.

Ghosh et al. [9] propose symbolic simulation in order to produce a set of Boolean functions which represent conditions for switching at each gate in the circuit. Given input switching rates, the switching probability at each gate is calculated by performing a linear traversal of the Binary Decision Diagrams (BDDs) representation of the corresponding Boolean function [14]. A general delay model which correctly computes the Boolean conditions that cause glitchings is used and correlations due to the reconvergence of input signals are taken into account.

Recently, researches on synthesis for low power have been carried out. Shen et al, [19] present algorithms for reducing power consumption during technology independent phase of logic synthesis. Intermediate nodes 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 consumption.

Prasad et al. [16] 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. Roy et al. [17] propose a low power state assignment method which use simulated annealing to find the state encoding of a finite state machine which the total probability weighted Hamming distance of the states are minimized.

Vaishnav et al. [21] tackle the low power synthesis problem in physical domain. They propose a low power performance driven placement procedure which minimizes the length, hence the capacitance loading, of the high switching nets and at the same time satisfying the delay constraints.

All the above power estimation tools and low power synthesis and design algorithms are based on a simple power consumption model where power is consumed only during the charging and discharging of the external capacitances driven by the gates. This assumption surely underestimates the power consumption of the circuit since it does not account for the power consumption due to the charging and discharging of the source/drain capacitance of the internal nodes of the gates. Power consumption due to internal capacitances contributes a significant portion of the total power consumption in the circuit (see Section 5). Any power estimation tool ignoring this factor will thus be inaccurate.

In this paper, we propose a new power consumption model which considers the internal power consumption and then present a technique to estimate the average internal power

consumption for dynamic DOMINO CMOS as well as static CMOS circuits. We also propose a low power technology decomposition procedure which produces a decomposed network with minimum switching activities and a low power technology mapping algorithm which hides the high switching nodes inside gates and reduces the fanout load driven by high switching activity nodes.

1.2 Calculation of Switching Probabilities

In what follows, we assume a *zero delay model* where gate delays are assumed to be zero [9] and thus ignore signal transitions due to glitching. Primary inputs are assumed to be uncorrelated (issues regarding the use of a real delay model and correlated primary inputs are discussed in Section 6).

For *n-type* dynamic circuits, the output is precharged to 1 and hence the switching probability is given by

$$P_{o_{1->0}} = P_{o=0} \tag{1}$$

where $P_{o=0}$ is the probability of signal o assumes value 0.

For *p-type* dynamic circuits, the output is predischarged to 0 and hence the switching probability is given by

$$P_{o_{0->1}} = P_{o=1} \tag{2}$$

where $P_{o=1}$ is the probability of signal o assumes value 1.

For static circuits, we assume that the present input signal value is independent of the previous value. Hence, switching probabilities are given by

$$P_{o_{0->1}} = P_{o=0} P_{o=1} \tag{3}$$

$$P_{o_{1(0->1)}|o_{2(0->1)}} = P_{o_1=0|o_2=0} P_{o_1=1|o_2=1} \tag{4}$$

where $P_{o_1=x|o_2=y}$ is the probability that signal o_1 assumes x given that signal o_2 assumes y. The signal probability at the output of a node is calculated by first building an Ordered Binary-Decision Diagrams (OBDD) [3] corresponding to the global function of the node and then performing a linear traversal of the OBDD using the procedure given in [14].

1.3 Organization of the Paper

The rest of this paper is organized as follows. In Section 2 we review the conventional power consumption models and present an extended model which accounts for the internal power consumption. We also introduce the notion of charging and discharging functions for an internal node of a gate and present techniques which use these functions to estimate the internal power consumption. In Section 3, we describe a procedure for constructing a NAND-decomposed network which has minimum total switching activity. Section 4 presents a technology mapping paradigm for low power which exploits the power/delay tradeoff curves to generate a mapped network with minimum power consumption subject to given timing constraints. Experimental results are presented in Section 5. We discuss some issues and extensions in Section 6 and give conclusions in Section 7.

2 Power Consumption Modeling and Analysis

2.1 A Simple Power Consumption Model

Power consumption in CMOS circuits is caused by three sources: the charging and discharging of capacitive loads during output switchings, the short circuit current which flows during output transitions and leakage current. The last two sources can be made small with proper device and circuit design techniques [22]. We therefore concentrate on the dynamic power consumption.

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

$$P_{avg}(g) = 0.5 \frac{V_{dd}^2}{T_{cycle}} C_{load} E(switching) \tag{5}$$

where

$$C_{load} = \sum_{j \in fanout(g)} C_G^j + C_{wire} \tag{6}$$

and C_G^j is the gate capacitance of j, C_{wire} is the wiring capacitance of the output net, V_{dd} is the supply voltage, T_{cycle} is the clock cycle time, and $E(switching)$ is the expected number of transitions at the output of g per clock cycle (See Section 1.2).

$E(switching)$ 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.

Figure 1: Charging and discharging of an internal node of a 2-input NOR gate.

2.2 Power Consumption at Internal Nodes of a Gate

In a typical 0.8μm technology, the diffusion capacitance is about 20% of the gate capacitance [1]. The charging and discharging in the source/drain capacitance of transistors in a gate thus makes a considerable contribution to the power consumption of the gate. In particular, note that the internal capacitances may be charged and discharged without any change in the gate output. This is illustrated in Figure 1. For a two input NOR gate, if the inputs are changing from 01 to 10, the output remains unchanged. However, when the inputs are 01, the capacitance of node A is charged which is then discharged when the inputs are changed to 10. This contributes to the power consumption in the circuit. The amount of power consumption depends on the frequency of charging and discharging and the source/drain capacitances (hence sizes of the transistors).

The power consumption at the internal node n of gate g in a static circuit is given by

$$P_{avg}^{int}(n) = 0.5\frac{V_{dd}^2}{T_{cycle}} C_{SD}^n E_n(charge/discharge) \tag{7}$$

where C_{SD}^n is the source/drain capacitance at n, and $E_n(charge/discharge)$ is the expected number of charge and discharge events at n per clock cycle.

C_{SD}^n and $E_n(charge/discharge)$ for internal nodes of gates in dynamic circuits is somewhat different and deserves more explanation. We consider *n-type* dynamic circuits. Similar results can be derived for *p_type* dynamic circuits. Here, output nodes are charged to 1 during the precharge period while all the gate inputs which are fed from the previous stage are maintained at 0. Therefore only the precharged output nodes

Figure 2: Charge sharing in dynamic circuits.

have direct charging path to V_{dd}. Consider Figure 2, during the evaluation period, if i_1 is 1 and i_2 is 0, there will be charge sharing between output node *out* and internal node n and the voltage seen at n will be equal to $\frac{C_{out}}{C_{out}+C_n}V_{dd}$.

The power consumption due to charge sharing is thus equal to

$$\begin{aligned} P_{avg}^{int}(n) &= 0.5\frac{V_{dd}^2}{T_{cycle}}\left(\frac{C_{out}}{C_{out}+C_n}\right)^2 C_n E_n(charge/discharge) \\ &= 0.5\frac{V_{dd}^2}{T_{cycle}} C_{eff}^n E_n(charge/discharge) \end{aligned} \quad (8)$$

where $C_{eff}^n = (\frac{C_{out}}{C_{out}+C_n})^2 C_n$ is the effective capacitance seen at n.

The effective capacitance of an internal node depends on how many internal nodes are connected to the precharged node at the same time. In Figure 2, if i_2 is also 1, then m is connected to n and *out*. The effective capacitance seen at n is thus smaller and is given by $C_{eff}^n = \left(\frac{C_{out}}{C_{out}+C_n+C_m}\right)^2 C_n$.

In general, the effective capacitance of a node can be approximated as

$$C_{eff}^n = \left(\frac{C_{out}}{C_{out}+\sum C_j}\right)^2 C_n \quad (9)$$

where C_j is the source/drain capacitance of internal node j which lies on the path from n to the precharged node. Charge sharing creates a drop in the output voltage and if

this drop is greater than a threshold voltage, it may create a logic error or loss of noise margin in the subsequent stages. Therefore for a dynamic circuit to operate properly, the voltage drop due to charge sharing must be small. This implies that $C_{out} \gg \sum C_i$, and $C^n_{eff} \approx C_n$.

When the charge sharing problem is severe, designers often precharge the internal nodes of the pulldown NFET chain for *n-type* circuits. In this case, for each precharged node n, since it is precharged every cycle, $E_n(charge/discharge)$ is equal to $P_{low}(n)$.

2.3 An Extended Power Consumption Model

To capture the power consumption due to internal capacitances, equation (5) must be augmented as follows:

$$P_{avg}(g) = 0.5\frac{V_{dd}^2}{T_{cycle}}\left(C_{load}E(switching) + \sum_{i \in internal\_nodes(g)} P^{int}_{avg}(i)\right) \tag{10}$$

where $P^{int}_{avg}(i)$ is given by either equation (7) or equation (8).

2.4 Calculation of Expected Number of Charge/Discharge Events

The internal power consumption depends on C^i_{SD} which is determined by the size of transistors in the gate, and $E_i(charge/discharge)$ which depends on the global function of i in terms of the primary inputs, the switching probabilities of the primary inputs, and the internal structure of the gate.

The main theorem for calculating the $E_i(charge/discharge)$ is given next.

Theorem 2.1 *The expected number of charge/discharge events per clock cycle at some internal node i which is not precharged (or predischarged) is given by*

$$E_i(charge/discharge) = \frac{P_{high}(i)P_{low}(i)}{P_{high}(i) + P_{low}(i)} \tag{11}$$

where $P_{high}(i)$ and $P_{low}(i)$ denote probabilities of i being charged to V_{dd} and discharged to Gnd.

Proof We find $E(n)$, the expected number of patterns $h(f^*)l$ in a string of characters consisting of n elements from the alphabet $\{h, f, l\}$ where h, f, l denote the input combinations which give rise to charging, floating, and discharging conditions at node i respectively, and f^* means zero or more f characters.

$E(n)$ is composed of two parts. The first part is the expected number of patterns $h(f^*)l$ in a string of $n-1$ characters (i.e. $E(n-1)$). The second part is the product of $P(h)$ and the probability $P((f^*)l, n-1)$ that a string with $n-1$ characters begins with pattern $(f^*)l$. The reasoning for this is that if the first character is an l or f, it will not create any pattern in addition to those included in $E(n-1)$; If the first character is an h, then any string of $n-1$ characters that begins with pattern $(f^*)l$ will add one additional $h(f^*)l$ pattern to $E(n)$. Since

$$\begin{aligned} P((f^*)l, n-1) &= P(l) + P(f)P(l) + \ldots + P(f)^{n-2}P(l) \\ &= P(l)\left(\frac{1-P(f)^{n-1}}{1-P(f)}\right), \end{aligned}$$

$E(n)$ is given by the following recurrence formula

$$\begin{aligned} E(n) &= E(n-1) + P(h)P((f^*)l, n-1) \\ &= E(n-1) + P(h)P(l)\left(\frac{1-P(f)^{n-1}}{1-P(f)}\right). \end{aligned}$$

This recurrence formula can be solved to yield

$$\begin{aligned} E(n) &= \frac{P(h)P(l)}{1-P(f)}\sum_{i=1}^{n}(1-P(f)^{i-1}) \\ &= \frac{P(h)P(l)}{1-P(f)}\left(n - \frac{1-P(f)^n}{1-P(f)}\right) \end{aligned}$$

Each character represents an input vector, and n input vectors means n cycles have elapsed. So the expected number of charge/discharge events per clock cycle is given by

$$E_i(charge/discharge) = \frac{P_{high}(i)P_{low}(i)}{1-P_{floating}(i)} \tag{12}$$

as n approaches ∞. Because

$$P_{high}(i) + P_{low}(i) + P_{floating}(i) = 1, \tag{13}$$

this is equal to

$$\frac{P_{high}(i)P_{low}(i)}{P_{high}(i)+P_{low}(i)}. \tag{14}$$

■

In the following, we show how to calculate $P_{high}(i)$ and $P_{low}(i)$.

For both dynamic and static circuits, $P_{high}(n)$ and $P_{low}(n)$ are given by

$$P_{high}(n) = P(F_c(n)) \tag{15}$$

$$P_{low}(n) = P(F_d(n)) \tag{16}$$

where $F_c(n)$ and $F_d(n)$ are Boolean functions representing conditions for n to be charged and discharged, respectively.

Let $TG(N,E)$ be *transistor graph* of gate g where each node represents an internal node of the gate, V_{dd} and Gnd and each edge represents a transistor of the gate. For static CMOS gates, two TGs are formed, one for the NMOS transistors and one for the PMOS transistors. For dynamic circuits only one NMOS (or PMOS) TG is formed depending on the logic type used. At the same time, V_{dd} (or Gnd) is replaced by the precharged (or predischarged) node. For the NMOS TG, each edge is labeled with the signal name of the gate of the corresponding n-transistor. For the PMOS TG, each edge is labeled with the inverted gate signal of the corresponding p-transistor (Figure 3). From the transistor graph, logic functions which charge/discharge the internal nodes can be obtained as stated below.

Theorem 2.2 *Let O denote the gate output node, G be the logic function of O, $paths(n_1,n_2)$ denote the set of all simple paths connecting n_1 and n_2 in $TG(N,E)$, and $L(P) = x_1 \wedge x_2 \wedge \ldots \wedge x_n$ where P is a simple path consisting of edges $e_1, e_2, \ldots, e_n$ and x_i is the label of edge e_i. The charging and discharging functions for internal node n are given as follows*

n-type dynamic

$$F_{c_N}^{dynamic}(n) = G \wedge \bigvee_{P \in paths(n, precharged\_node)} L(P), \tag{17}$$

$$F_{d_N}^{dynamic}(n) = \bigvee_{P \in paths(n,Gnd)} L(P) \tag{18}$$

p-type dynamic

$$F_{c_P}^{dynamic}(n) = \bigvee_{P \in paths(n,V_{dd})} L(P), \tag{19}$$

$$F_{d_P}^{dynamic}(n) = \overline{G} \wedge \bigvee_{P \in paths(n, predischarged\_node)} L(P) \tag{20}$$

static NMOS

$$F_{c_{NMOS}}^{static}(n) = G \wedge \bigvee_{P \in paths(n,O)} L(P), \tag{21}$$

$$F_{d_{NMOS}}^{static}(n) = \bigvee_{P \in paths(n,Gnd)} L(P) \tag{22}$$

static PMOS

$$F_{c_{PMOS}}^{static}(n) = \bigvee_{P \in paths(n,V_{dd})} L(P), \tag{23}$$

$$F_{d_{PMOS}}^{static}(n) = \overline{G} \wedge \bigvee_{P \in paths(n,O)} L(P) \tag{24}$$

where $\wedge$ *and* $\vee$ *denote Boolean* AND *and* OR *operations.*

Proof In the following we present the proof for cases where internal node n is in an *n-type* dynamic circuit or in the PMOS part of a static CMOS gate. Proofs for other cases are similar.

Consider a path P in $TG(N, E)$ connecting n_1 and n_2. If the input signal vector satisfies $L(P)$, n_1 and n_2 are connected. For *n-type* dynamic circuits, internal node n is charged up when there is a path from the *precharged node* to n during the evaluation period, and at the same time the output node is not discharged. The logic function representing logic conditions for the connection between the *precharged node* and the internal node n is equal to the disjunction of all the input signal vectors that create a connected path between n and the *precharged node*. The logic function representing the condition that the output node is not discharged is simply G. Therefore, we have

$$F_{c_N}^{dynamic}(n) = G \wedge \bigvee_{P \in paths(n, precharged\_node)} L(P). \tag{25}$$

n is discharged when there is a path connecting n to Gnd and hence the discharging function is given by

$$F_{d_N}^{dynamic}(n) = \bigvee_{P \in paths(n,Gnd)} L(P). \tag{26}$$

For an internal node n in the PMOS part of the static CMOS gate, the logic conditions for having a charging path from V_{dd} to n is equal to the disjunction of all the input signal vectors that create a connected path between n and V_{dd}. Hence

$$F_{c_{PMOS}}^{static}(n) = \bigvee_{P \in Path(n,V_{dd})} L(P). \tag{27}$$

Figure 3: The transistor graph and charging/discharging functions for a 3-input NOR gate.

An internal node n in the PMOS part has a discharging path when there exists a path connecting n to the gate output and a path connecting the gate output to the Gnd. Logic conditions which connect n to the gate output are given by $\bigvee_{P \in Path(n,O)} L(P)$ while the logic condition which give rise to a path connection between the gate output and Gnd is the complement of the gate's logic function ($\overline{G}$). Thus

$$F^{static}_{d_{PMOS}}(n) = \overline{G} \wedge \bigvee_{P \in Path(n,O)} L(P). \tag{28}$$

■

Figures 3 shows an example of the charging and discharging functions for a 3-input NOR gate.

3 Power Efficient Technology Decomposition

It is difficult to come up with a decomposed network which will lead to a minimum power consumption implementation after power-efficient technology mapping is applied since gate loading and mapping information are unknown at this stage. Nevertheless, we have observed that a decomposition scheme which minimizes the sum of the switching activities at the internal nodes of the network, is a good starting point for power-efficient technology mapping. We illustrate this point with a simple example (Figure 4a). A four-input AND gate can be decomposed into a tree of 2-input AND gates in two ways. These two decompositions have different total switching activities. Assuming *n-type* dynamic circuit and independent inputs, let $P(a)$=0.7, $P(b)$=0.3, $P(c)$=0.3 and $P(d)$=0.7. The total switching activities for configurations A and B are 2.317 and 2.464, respectively. Configuration A appears to be better than configuration B since the sum of the switching activities at its internal nodes is smaller. Furthermore, if we assume that the cell library has 2-input and 3-input AND gates, the minimum power mapping with a power value of 2.107 is obtained from configuration A (Figure 4b). In this example, decomposition with lower switching activity leads to mapping with lower power consumption.

We denote the problem of generating a NAND-decomposed network with minimum total switching activity as the MINPOWER decomposition. The performance-oriented version of the above problem requires that the increase in the height of the decomposed network (compared to the undecomposed network) be bounded. We refer to this problem as the BOUNDED-HEIGHT MINPOWER decomposition.

3.1 Tree Decomposition

We describe algorithms for solving the MINPOWER decomposition for a fanout-free logic function (i.e. a function that has a tree realization).

The basic algorithm is similar to Huffman's algorithm [10] for constructing a binary tree with minimum average weighted path length. We denote the leaves of a binary tree by $v_1, v_2, \ldots, v_n$, the "path length" from the root to v_i by l_i, and the weight of leaf v_i by w_i. Assuming that the root is at level zero (the highest level), leaf v_i is at level l_i. Given a set of weights w_i, there is a simple $O(nlogn)$-time algorithm due to Huffman for constructing a binary tree such that the cost function $\sum_{i=1}^{n} w_i l_i$ is minimum.

In a more general setting, a *weight combination function* $F(x, y)$ (which is any symmetric function chosen as a binary operator on the weight space U) is used to produce the weight W of internal nodes during tree construction. For each tree T, a *tree cost function*

Figure 4: An example showing the effect of technology decomposition on total switching activity.

$G(W_1, W_2, \ldots, W_{n-1})$ gives the cost.[1] Parker [11] characterized a wide class of weight combination functions, for which Huffman's algorithm produces optimal trees under corresponding tree cost functions. We give some definitions first and then state Parker's main theorem.

Definition 3.1 *A weight combination function F is quasi-linear if $F(x,y) = \phi^{-1}(\lambda\phi(x) + \lambda\phi(y))$ where λ is a nonzero constant and ϕ is a real-valued invertible function on the weight space U. (Note F is symmetric, and conjugate under ϕ to the linear map $\lambda(x+y)$.)*

Definition 3.2 *A tree cost function G: $U^{n-1} \rightarrow R$ is* Schur concave *if*

$$(x_i - x_j)\left(\frac{\partial G}{\partial x_i} - \frac{\partial G}{\partial x_j}\right) \leq 0 \tag{29}$$

for all $x_i, x_j \in U$, $i, j \in 1, \ldots, n-1$.

Theorem 3.1 *[11] If the weight combination function F is quasi-linear and the corresponding function ϕ is convex, positive, (or concave, negative) and strictly monotone*

[1] In Huffman's original paper, $F(x,y) = x + y$ and $G = \sum_{i=1}^{n-1} W_i$. It is easy to show that $G = \sum_{i=1}^{n} w_i l_i$.

and $\lambda \geq 1$, then the Huffman tree will have the least cost when G is any Schur concave *function of the internal node weights.*

If $F(x, y)$ does not satisfy the above conditions, then Huffman's algorithm may not produce the optimal solution. We propose the following $O(n^2 logn)$ greedy algorithm to solve the decomposition problem for general weight combination functions.

Algorithm 3.1 *For every pair w_i and w_j of the n non-negative weights $w_1, w_2, \ldots, w_n$, compute $F_{ij}(w_i, w_j)$ and store in list L . Find the smallest F_{ij}, say F_{12}. Replace the two nodes by a single node having the weight $W_1 = F_{12}$ and two sons with weight w_1 and w_2. Eliminate all $F_{1k}(w_1, w_k)$ and $F_{2k}(w_2, w_k)$ from L. Compute $F_{1j}(W_1, w_j)$ and insert it into L. Do this recursively for the $n-1$ weights $W_1, w_3, \ldots, w_n$. The final single node with weight W_{n-1} is then the root of the binary tree.*

To solve the MINPOWER tree decomposition problem, we must use appropriate weight combination and tree cost functions. We thus distinguish among two cases as follows.
Dynamic Circuits
For (*n-type* circuits), dynamic gate outputs are precharged to 1 and switching occurs when the output changes to 0 during the evaluation phase. For a 2-input AND gate composed of a 2-input NAND gate and a static inverter (Figure 5a), the inverter output is 0 during the precharge period and the switching probability (without input signal correlations) is given by

$$W_o = w_{i_1} w_{i_2} \tag{30}$$

where W_o and w_{i_x} values are probabilities of the output and inputs assuming value 1.

For *p-type* circuits, dynamic gate outputs are predischarged to 0 and transition occurs when output evaluates to 1. The corresponding formula for the switching probability for a 2-input AND gate composed of a 2-input NAND gate and a static inverter (Figure 5b) is then given by

$$W_{\overline{o}} = 1 - (1 - w_{\overline{i_1}})(1 - w_{\overline{i_2}}) \tag{31}$$

where the $W_{\overline{o}}$ and $w_{\overline{i_x}}$ values are probabilities of the output and inputs assuming value 0.

The weight combination functions $F(w_{i_1}, w_{i_2}) = W_o$ or $F(w_{\overline{i_1}}, w_{\overline{i_2}}) = W_{\overline{o}}$ are used during the AND decomposition. The corresponding tree cost function G is given by

$$G = \sum_{i=1}^{n-1} W_i. \tag{32}$$

Lemma 3.2 *W_o and $W_{\overline{o}}$ given in (30) and (31) above are quasi-linear functions.*

Figure 5: 2-input n-type and p-type dynamic AND gates.

Proof For (30), since w_i is within the range of [0,1] we can take $\phi(x) = -log(x)$ which is a convex, positive and decreasing function and λ= 1. This shows that W_o is quasi-linear.

Similarly, for (31), since $w_{\bar{i}}$ is within the range of [0,1] we can take $\phi(x) = -log(1-x)$ which is a convex, positive and increasing function and λ= 1. This shows that $W_{\bar{o}}$ is quasi-linear. ■

Lemma 3.3 *G given in (32) above is* Schur concave.

Proof Since $\frac{\partial G}{\partial W_i} = \frac{\partial G}{\partial W_j}$, $(W_i - W_j)\left(\frac{\partial G}{\partial W_i} - \frac{\partial G}{\partial W_j}\right) = 0$, G is *Schur concave.* ■

Theorem 3.4 MINPOWER *tree decomposition for dynamic* CMOS *circuits with uncorrelated input signals can be solved optimally by Huffman's algorithm using the weight combination functions (30) and (31).*[2]

Proof Follows from Lemma 3.2 and 3.3. ■

If the input signals to the AND gate are correlated, (30) and (31) cannot be used as the weight combination function. The switching probabilities for *n-type* and *p-type* circuits are then given by

$$W_o = w_{i_1} w_{i_2|i_1}, \tag{33}$$

[2]Indeed, a chain-like tree decomposition will be obtained.

$$W_{\overline{o}} = 1 - (1 - w_{\overline{i_1}})(1 - w_{\overline{i_2}|i_1}) \tag{34}$$

respectively where $w_{i_2|i_1}(w_{\overline{i_2}|i_1})$ is the conditional probability of i_2 ($\overline{i_2}$) given i_1.

Lemma 3.5 *W_o and $W_{\overline{o}}$ given in (33) and (34) are not quasi-linear functions.*

Proof We present the proof for (33). Proof for the other case is similar.

One criterion for a function F to be quasi-linear is increasingness, i.e. $F(u,x) \leq (\geq) F(u,y)$ if $x \leq (\geq) y$ [11]. However, in (33), we could have $w_{i_2} \leq w_{i_3}$, yet $w_{i_2|i_1} > w_{i_3|i_1}$, thus $F(w_{i_1}, w_{i_2}) > F(w_{i_1}, w_{i_3})$. W_o does not satisfy the increasingness property and hence is not quasi-linear. ■

Since W_o's given in equations (33) and (34) are not quasi-linear, hence we use Algorithm 3.1 to solve the decomposition problem.

Static Circuits

For static CMOS circuits, we need to minimize sum of the probabilities for output switchings from 0 to 1 and 1 to 0. Thus, the weight combination function W_o for a 2-input AND gate is equal to $W_{o_{0->1}} + W_{o_{1->0}}$ where

$$W_{o_{0->1}} = w_{i1_{0->1}} w_{i2_{0->1}} + w_{i1_{1->1}} w_{i2_{0->1}} + w_{i1_{0->1}} w_{i2_{1->1}}, \tag{35}$$

$$W_{o_{1->0}} = w_{i1_{1->1}} w_{i2_{1->0}} + w_{i1_{1->0}} w_{i2_{1->1}} + w_{i1_{1->0}} w_{i2_{1->0}}. \tag{36}$$

If input signals are correlated, conditional probabilities between the input signal transitions have to be used in order to calculate the output transition probability. $W_{o_{0->1}}$ and $W_{o_{1->0}}$ are then given by

$$W_{o_{0->1}} = w_{i1_{0->1}} w_{i2_{0->1}|i1_{0->1}} + w_{i1_{1->1}} w_{i2_{0->1}|i1_{1->1}} + w_{i1_{0->1}} w_{i2_{1->1}|i1_{0->1}}, \tag{37}$$

$$W_{o_{1->0}} = w_{i1_{1->1}} w_{i2_{1->0}|i1_{1->1}} + w_{i1_{1->0}} w_{i2_{1->1}|i1_{1->0}} + w_{i1_{1->0}} w_{i2_{1->0}|i1_{1->0}}. \tag{38}$$

Note that for static circuits, every internal node is assigned multiple weights (i.e. $w_{i_{0->1}}$, $w_{i_{1->0}}$, $w_{i_{1->1}}$). The weight combination uses all these weights through equations (35) to (38). This general class of problems cannot be solved using the Huffman's algorithm which applies when each internal node is given a single weight. Therefore, we resort to Algorithm 3.1.

3.2 Bounded-Height Tree Decomposition

Here, the objective is to construct a MINPOWER binary tree for a given list of weights (signal switching probabilities) with the restriction that the height of each weight (defined as $max_i\ l_i$) does not exceed a given integer L. The best known algorithm for solving BOUNDED-HEIGHT MINSUM problem is an $O(nL)$ algorithm due to Larmore and Hirschberg [13]. Their approach (based on the PACKAGE-MERGE algorithm) transforms the BOUNDED-HEIGHT MINSUM tree decomposition problem to an instance of the COIN COLLECTOR's problem.[3] The PACKAGE step in this algorithm uses the Huffman's Algorithm to merge two items at level i to form a new item at level $i+1$. The MERGE step merges the newly formed items with the original items at each level. Details of the algorithm can be found in [13].

For the general weight combination functions, the Larmore-Hirschberg's algorithm has to be modified as follows: In the PACKAGE step, replace the Huffman's algorithm by Algorithm 3.1; the MERGE step is unchanged. Using the modified Larmore-Hirschberg's algorithm, the BOUNDED-HEIGHT MINPOWER tree decomposition can be solved heuristically for the general weight combination functions.

4 Power Efficient Technology Mapping

The problem of technology mapping for low power can then be stated as follows: 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. A successful and efficient solution to the minimum area mapping problem was suggested in [12] and implemented in programs such as DAGON and MIS. 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.

Traditional goal for technology mapping is to minimize the total area (or delay) of the mapped circuit. In [5], a near-optimal solution is presented for finding the minimum area solution under delay constraints. Their approach consists of two steps: In the first step, delay functions (which capture arrival time-gate area tradeoffs) at all nodes in the network are generated. In the second step a mapping solution based on the computed

[3] An instance(I, X) of the COIN COLLECTOR's problem is defined as given a set I of m items each of which has a *width* and *weight*, find a subset of S of I whose widths sum to X and has minimum total weight.

delay functions and the required times at the primary outputs is found. For a NAND-decomposed tree, subject to load calculation errors, this two-step approach finds the minimum area mapping satisfying all delay constraints if such solution exists.

Our low power technology mapper follows a procedure similar to the above, with the difference that the objective is to minimize the sum over all gates of the average power consumed in the mapped network subject to given required time constraints. The approach also consists of two steps: First a postorder traversal is used to determine a set of possible arrival times and accumulated power consumptions at each node of the network. Once the user specifies a single required time, a second preorder traversal starting from the primary outputs is performed to determine the mapping solution that minimizes the average power subject to the required time constraints.

4.1 Terminology

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 (that is, $n = n_1' + n_2'$ if n has two inputs or $n = n_1'$ if n has a single input). 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 $n_i \in inputs(n, g)$. Note that given node n and gate g matching at n, $inputs(n, g)$ are uniquely determined. However, n_i may have many distinct mapped-parents.

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 consumed in its mapped transitive fanin cone (but excluding the power consumed at the load driven by the node which has yet to be decided by a later mapping). In addition to the power and delay values, the matching gate and input bindings for the match are also stored with each point on the curve. Points on the curve represent various mapping solutions with different tradeoffs between average power and speed. A point (t_1, p_1) is inferior if there exists another point (t_2, p_2) on the curve such that either $p_1 > p_2$ and $t_1 \geq t_2$ or $p_1 \geq p_2$ and $t_1 > t_2$ (Figure 6). All inferior points are dropped from the power-delay curve.

4.2 Arrival time and Power Cost Calculation

We have adopted the pin-dependent SIS library delay model for the calculation of the arrival time and power consumption. Suppose that gate g has matched at node n, then

Figure 6: Inferior and non-inferior points on a power-delay curve.

the output arrival time at n is given by

$$A(n, g, C_n) = max_{n_i \in inputs(n,g)}(\tau_{i,g} + R_{i,g}C_n + A(n_i, g_i, C_{n_i})) \tag{39}$$

where $\tau_{i,g}$ is the *intrinsic* gate delay from input i to output of g, $R_{i,g}$ is the *drive* resistance of g corresponding to a signal transition at input i, C_n is the load capacitance seen at n, $A(n_i, g_i, C_{n_i})$ is the arrival time at input i corresponding to load C_{n_i} seen at that input, and g_i is the best match found at input i.

Under the simple power consumption model, the power cost for gate g matching at node n is given by

$$P_{avg}(n, g) = \sum_{n_i \in inputs(n,g)} (0.5 C_{n_i} \frac{V_{dd}^2}{T_{cycle}} E_{n_i} + P_{avg}(n_i, g_i)) \tag{40}$$

where E_{n_i} is the expected switching activity at node n_i, and $P_{avg}(n_i, g_i)$ is the accumulated power cost at n_i assuming a gate matching g_i.

The input to the technology mapper is a 2-input NAND(NOR) decomposed network of the optimized network from the technology independent phase. The signal probability $P(n)$ of each node n of the decomposed network is calculated prior to the mapping.

Under the extended power consumption model, the power cost is given by

$$P_{avg}(n, g) = \sum_{j \in internal\_nodes(g)} P_{avg}^{int}(j) + \sum_{n_i \in inputs(n,g)} (0.5 C_{n_i} \frac{V_{dd}^2}{T_{cycle}} E_{n_i} + P_{avg}(n_i, g_i)) \tag{41}$$

where $P_{avg}^{int}(j)$ is given by equation (7).

The typical gate library model contains information such as input capacitance for each input pin, pin-dependent intrinsic gate delay, and the pin-dependent output drive. To support the calculation of the internal power consumption, we add to this data the following information: a) the source/drain capacitance for each internal node; b) the charging and discharging functions in terms of the gate inputs for each internal node.

When calculating the power cost at node n, the power contribution from the output load driven by n is not included. We however have the following lemma for the power cost calculation:

Lemma 4.1 *The power cost of a match g at node n given by equation (41) is the exact power cost for a zero delay model.*

Proof Under a zero delay model, the switching activity at node n only depends on the global function of the node in terms of the circuit primary inputs and not its particular implementation. Hence, E_n is independent of the gate matching at n. Furthermore, note that since the power consumption due to output load of n is calculated only when the fanout gate of n is known, equation (41) is not subject to the *unknown load problem*. ■

The average power contribution of the n's output load will be included at *mapped-parent(n)*. When the mapping reaches a primary output, every point on the power-delay curve has a *power* value equal to the total average power consumption of the mapped tree minus the power consumption at the primary output load. The power consumption at the primary output load depends on E_n at the output and the load capacitance connected to it which are both *independent* of the mapping configuration. Hence, it only causes a fixed shift of the curve along the power axis. It does not affect the selection of the optimal point from the power-delay curve.

4.3 Tree Mapping

In this section, we focus on tree mapping. Later, we shall describe the extension to DAG mapping. In particular, we describe two tree-traversal operations which are applied to a NAND-decomposed tree in order to obtain a technology mapping solution which minimizes the average power consumption while satisfying the timing constraints.

4.3.1 Tree Traversals

On the first traversal, we begin at the leaf nodes of the NAND-decomposed tree. Since each node n possesses a set of possible arrival time - average power points which are reflected in its power-delay function, the power-delay function at any *mapped-parent*(n) must also reflect these possible arrival time - average power tradeoffs. A postorder traversal of the NAND-decomposed tree is performed, where for each node n and for each gate g matching at n, a new power-delay function is produced by appropriately merging the power-delay functions at the $inputs(n, g)$. Merging must occur in the common region in order to ensure that the resulting function reflects feasible matches at the $inputs(n, g)$. The power-delay functions for successive gates g matching at n are then merged by applying a *lower-bound merge* operation on the corresponding power-delay functions (see [5] for details of these operations). At a given node n, the resulting power-delay function will describe the arrival time - average power tradeoffs in propagating a signal from the tree inputs to the output of n.

The second traversal begins when the mapping reaches the root(primary output). The user is allowed to select the arrival time - average power tradeoff which is most suitable for his application. Given the required time t at the root of the tree, a suitable (t, p) point on the power-delay function for the root node is chosen. The gate g matching at the root which corresponds to this point and $inputs(root, g)$ are, thus, identified. The required times t_i at $inputs(root, g)$ are computed from t and g. The preorder traversal resumes at $inputs(root, g)$ where t_i is the constraining factor and a matching gate g_i with minimum power consumption satisfying t_i is sought.

4.3.2 Timing Recalculation

The gate delay is a function of the load it is driving. During the postorder tree traversal, the output of current node n_i, is not mapped hence the load capacitance is unknown (unless, all the gates in the library have identical pin capacitances). At this time the load value is assumed to be that offered by the smallest two-input NAND gate in the library. When we come to a node $n =$*mapped-parent*(n_i) with matching gate g, we know the exact load seen by n_i. This load is equal to the input capacitance of g and is, in general, different from the default load. Therefore, in order to calculate the arrival time at node n, the output arrival times for all nodes in $inputs(n, g)$ must be adjusted to account for the change in the load capacitance. Similarly, during the preorder tree traversal, when a gate g is selected to match at n, the load seen by $inputs(n, g)$ must be recalculated.

In order to account for this load change (Δ_i), the power-delay curves at the inputs have

to be appropriately shifted. In particular, since the drive resistance of gate matching at n_i and giving rise to a point p_j on power-delay curve of n_i is stored with that point, the delay shift is computed as $\Delta_i \times p_j.gate.drive$.

4.3.3 Optimality of the Tree Mapping Algorithm

The following theorem can be stated.

Theorem 4.2 *Under a zero delay model, the tree mapping algorithm finds the minimum power consumption solution for given delay constraints (subject to the error due to unknown loads during the arrival time calculation).*

Proof From Lemma 4.1, under a zero delay model, the power consumption calculation is exact and not subject to the *unknown load problem* and hence is similar in nature to the area calculation. The proof of optimality of the tree mapping algorithm is thus similar to that of the area-delay mapping algorithm. ■

4.4 DAG Mapping

Most of the real circuits are not trees, but general DAGs. The problem of mapping a DAG even for the unit fanout model is NP-hard [2]. Therefore, we resort to heuristics. During the power-delay curve computation step, nodes are visited in postorder. For each node, we compute the power-delay curve as in case of trees. However, if the input for a candidate match at the node is coming from a multiple fanout node, we divide the average power contribution of that input by the fanout count of the input node . By reducing the average power contribution, we favor a solution in which multiple fanout nodes are preserved after mapping, which reduces logic duplication and improves the final mapped average power. This heuristic which permits tree boundary crossing only for nodes with relatively few fanouts was also adopted by the MIS mapper [7]. During the gate selection step, if we come to a node which has already been mapped, we check if the mapped solution at the node satisfies the timing requirement. If so, we keep the mapping; otherwise, we replace it with another solution from the power-delay curve which satisfies the current timing requirement and has minimum average power.

4.5 Complexity Analysis

The technology mapper calculates the switching probabilities of each intermediate node of the network prior to the actual mapping process. The calculation employs OBDDs and hence the time complexity is equal to that of building OBDDs for the network.

For the mapping procedure, the complexity is similar to that of area-delay mapping given in [5] where it was shown that the complexity of generating power-delay curve for each candidate match g is $O(kNlog(N_{max}))$ where $N = \sum_{i=0}^{k} N_i$, $N_{max} = max_{i=0}^{k} N_i$, N_i denotes the number of points on the power-delay curve of input i of gate g, and k is the number of inputs.

For a finite size library, the number of points on the *power-delay* curve of n will remain linear in the total number of points in the power-delay curve of $inputs(n, g)$. Therefore, the number of points increases linearly from one level to another. If the tree is node-balanced, then the number of points will remain polynomial. If we bound the number of points to be generated for each power-delay curve to M, then the complexity becomes $O(kMlog(kM))$ which is a constant since the number of inputs for any gate in the library is bounded.

5 Experimental Results

Table 1 shows ratio of the internal power consumption (due to the parasitic source/drain capacitances) to the power consumption due to the external gate capacitances for a subset of ISCAS-89 and MCNC-91 benchmarks. It is seen that if internal power consumption is disregarded, then the power consumption will be underestimated by an average of 16.5%.

To evaluate the effectiveness of the technology decomposition and mapping for low power, we applied our algorithms on the above benchmarks. Static CMOS circuits were used in the experiments and all primary inputs were assumed to be pairwise and temporally independent. The delay was calculated using the augmented pin-dependent SIS library delay model as described in Section 4.2. We used an industrial library with 44 gates.

Table 2 shows the total switching activity of the decomposed networks for different technology decomposition methods. It is shown that the low power technology decomposition reduces the total switching activity in the networks by 5% over the conventional balanced tree decomposition method.

To study the effect of including internal power consumption in the power cost calculation during technology mapping, we applied *pd-map* using equation (40) (without internal power consumption) and then using equation (41) (with internal power consumption) and compiled the internal and total power consumption in Table 3. It can be seen that the internal power consumption and total power consumption are reduced by an average of 6% and 0.8%, respectively if equation (41) is used. We expect that for libraries which have more complex gates than the one we used, the percentage improvement will be higher. In the rest of the experiments, equation (41) was used for the power cost calculation.

Tables 4 and 5 contain our experimental results using different technology decomposition and mapping combinations. All methods have the same starting point, that is, circuits optimized by the SIS rugged script[18]. Method A uses area-delay mapping (*ad-map*) algorithm of [5] and methods B to D use power-delay mapping (*pd-map*). Methods A and B take in a NAND-decomposed network generated by the conventional balanced tree decomposition algorithm. Method C uses the MINPOWER technology decomposition (*minpower_t_decomp*) while method D uses the BOUNDED-HEIGHT MINPOWER decomposition (*bh_minpower_t_decomp*).

To see the impact of the *pd-map* on the average power consumption, we compare results of methods A and B. It is seen that with *pd-map*, the power consumption is improved by an average of 15.9% over *ad-map*. The active cell area is increased by an average of 12.2% while the circuit delay is improved by 2%.

To illustrate the impact of the (*minpower_t_decomp*) on the average power consumption, we compare the results of methods of B and C. The power consumption is improved by an average of 2.5% at the expense of 3.5% increase in area and 2.8% degradation in performance. From B and D, we see that *bh_minpower_t_decomp* improves the power consumption by an average of 1.6% over the conventional decomposition at the expense of 0.8% increase in area and 1.8% degradation in performance (Note that *bh_minpower_t_decomp* improves performance by an average of 1% over *minpower_t_decomp*). The best overall result is achieved when *pd-map* is applied after the network is decomposed using *minpower_t_decomp*. The average power consumption is reduced by an average 18% at the expense of 16% increase in area while the circuit delay is unchanged.

Comparing Tables 2 and 5, we observe that although the reduction in power consumption after mapping is not directly proportional to the reduction in the switching activities of the network before mapping, networks with lower switching activity often result in a mapped circuit with lower power consumption. We believe the small gain of *minpower_t_decomp* is due to the fact that most nodes in the optimized network are relatively simple due to the fast-extract and quick decomposition operations performed prior to the technology decomposition step. Therefore, the *minpower_t_decomp* does not

have much freedom in improving the power efficiency through NAND decomposition.

To see how power consumption is actually reduced by *pd_map*, we selected the s832 circuit and plotted the number of nets and the average loading on nets versus the switching rate. Figures 7 and 8 summarize the results. From Figure 7, we can see that *pd-map* tends to reduce the number of high switching activity nets at the expense of increasing the number of low switching activity nets. From Figure 8, we can see that for the remaining high switching activity nets, *pd-map* tries to reduces the average loading on nets. By doing these, *pd-map* minimizes the total weighted switching activities and hence the total power consumption in the circuit.

circuit	power consumption due to gate cap.	power consumption due to internal cap.	ratio of int. power consumption to ext. power consumption * 100
C1908	207.00	49.00	23.67
C432	92.00	18.00	19.57
alu2	132.00	31.00	23.48
apex7	107.00	17.00	15.89
cordic	34.50	5.30	15.36
example2	134.00	18.00	13.43
pair	760.00	128.00	16.84
parity	34.70	7.20	20.75
pm1	20.80	2.90	13.94
s208	34.90	6.20	17.77
s344	59.40	12.30	20.71
s382	75.00	11.80	15.73
s386	41.80	4.80	11.48
s400	75.80	12.20	16.09
s420	72.00	12.40	17.22
s444	76.40	12.40	16.23
s510	120.30	21.50	17.87
s526	95.90	15.40	16.06
s641	79.00	10.80	13.67
s713	80.30	10.70	13.33
s820	128.20	17.80	13.88
s832	122.70	18.50	15.08
ttt2	105.90	16.90	15.96
x1	147.30	20.50	13.92
x3	353.00	58.10	16.46
x4	168.40	25.90	15.38
avg.			16.53

Table 1: Internal power consumption versus external power consumption.

circuit	balanced decomposition	power eff. decomposition	% reduction in switching rate
C1908	672.00	657.00	2.23
C432	335.00	318.00	5.07
alu2	422.00	389.00	7.82
apex7	394.00	385.00	2.28
cordic	119.00	110.00	7.56
example2	473.00	457.00	3.38
pair	2628.00	2491.00	5.21
parity	110.00	110.00	0.00
pm1	75.00	70.00	6.67
s208	132.00	128.00	3.03
s349	207.00	206.00	0.48
s382	245.00	233.00	4.90
s386	125.00	109.00	12.80
s420	247.00	239.00	3.24
s400	275.00	258.00	6.18
s444	252.00	245.00	2.78
s510	410.00	392.00	4.39
s641	300.00	282.00	6.00
s713	309.00	284.00	8.09
s820	401.00	367.00	8.48
s832	385.00	352.00	8.57
ttt2	339.00	327.00	3.54
x1	499.00	453.00	9.22
x3	1224.00	1192.00	2.61
x4	598.00	584.00	2.34
avg.			5.08

Table 2: The total switching activity in the network for different decomposition schemes.

circuit	int. power for I	tot. power for I	int. power for II	tot. power for II	% red. int. power	% red. tot. power
C1908	48.02	254.80	49.36	256.52	-2.79	-0.68
C432	19.54	110.53	18.19	110.32	6.95	0.19
alu2	28.83	160.68	31.03	163.80	-7.64	-1.94
apex7	18.46	125.79	17.39	124.84	5.76	0.75
cordic	6.28	41.38	5.30	39.87	15.65	3.67
example2	19.53	153.95	17.94	152.73	8.15	0.79
pair	138.53	896.15	127.95	888.07	7.64	0.90
parity	10.09	46.84	7.15	41.91	29.12	10.52
pm1	2.90	23.36	2.92	23.77	-0.89	-1.78
s208	6.77	41.80	6.23	41.16	8.05	1.53
s349	12.24	71.35	11.82	71.76	3.47	-0.58
s382	12.57	87.77	11.79	86.79	6.27	1.11
s386	4.30	45.85	4.82	46.69	-11.99	-1.82
s400	13.40	89.16	12.15	88.01	9.33	1.29
s420	13.70	85.69	12.40	84.46	9.54	1.43
s444	13.30	89.38	12.40	88.82	6.74	0.63
s510	22.37	142.56	21.51	141.85	3.84	0.50
s641	11.72	90.73	10.80	89.87	7.84	0.95
s713	11.55	92.10	10.70	91.06	7.36	1.12
s820	19.34	146.62	17.80	146.02	7.99	0.41
s832	18.65	140.94	18.51	141.20	0.76	-0.19
ttt2	18.48	123.08	16.81	122.80	9.00	0.22
x1	22.90	168.99	20.51	167.82	10.44	0.69
x3	61.95	414.51	58.11	411.14	6.19	0.81
x4	27.30	193.51	25.81	194.30	5.45	-0.41
aver.					6.09	0.80

Table 3: Technology mapping results I: pd-map using equation (40), i.e. not considering internal power consumption, II: pd-map using equation (41), i.e. considering internal power consumption.

cir-	A			B		
cuit	power	area	delay	power	area	delay
C1908	282.54	6091.44	26.17	256.52	6945.86	25.36
C432	131.18	2278.00	30.73	110.32	2429.64	30.11
alu2	200.67	4266.32	26.05	163.80	5246.88	29.03
apex7	142.02	2734.96	11.69	124.84	2990.64	11.62
cordic	47.97	820.76	6.43	39.87	921.74	6.62
example2	176.77	3822.28	11.30	152.73	4371.04	9.76
pair	1001.34	18380.40	28.20	888.07	21583.88	29.86
parity	52.45	853.40	7.01	41.91	795.60	6.14
pm1	31.11	592.96	5.17	23.77	659.60	5.15
s208	49.06	867.68	8.14	41.16	1007.76	9.99
s349	83.00	1637.44	12.57	71.76	1753.72	13.14
s382	97.26	1878.16	10.42	86.79	2127.72	10.44
s386	64.88	1579.64	11.36	46.69	2031.84	11.71
s400	98.49	1829.88	10.77	88.01	2056.32	11.26
s420	98.39	1785.00	15.23	84.46	2139.96	18.39
s444	101.44	1819.68	10.97	88.82	1993.76	12.24
s510	164.13	2869.60	20.39	141.85	3376.88	15.78
s641	108.22	2137.92	21.12	89.87	2297.04	20.21
s713	108.47	2119.56	21.52	91.06	2264.40	20.01
s820	183.51	3423.12	12.59	146.02	3982.76	11.36
s832	178.53	3450.32	14.30	141.20	3941.28	10.98
ttt2	146.04	2528.92	12.26	122.80	2760.12	12.67
x1	204.66	3683.56	9.38	167.82	4025.60	8.14
x3	485.68	9200.40	13.45	411.14	10509.06	11.50
x4	227.05	4714.44	18.79	194.30	4833.44	15.68

Table 4: Technology decomposition and mapping results A : *ad_map* with balanced decomposition , B : *pd_map* with balanced decomposition.

cir-	C			D		
cuit	power	area	delay	power	area	delay
C1908	249.76	7226.36	26.84	254.41	7058.06	25.57
C432	106.39	2506.48	30.32	110.07	2456.84	29.12
alu2	161.28	5451.56	29.04	159.59	5375.40	28.50
apex7	123.89	3123.92	11.93	124.33	3090.94	11.52
cordic	37.65	866.32	8.26	38.80	879.92	7.24
example2	150.00	4522.68	10.21	150.92	4426.12	12.16
pair	867.31	21803.52	29.57	878.25	21896.68	29.75
parity	41.91	795.60	6.14	41.91	795.60	6.14
pm1	22.60	683.40	5.88	22.59	637.84	5.82
s208	40.54	1054.00	9.87	40.68	1054.68	10.42
s349	71.78	1754.40	13.10	72.14	1765.96	13.10
s382	85.09	2188.92	10.38	85.31	2086.24	10.38
s386	45.19	2114.12	11.16	45.18	2036.60	10.49
s400	86.73	2112.76	11.13	87.39	2034.56	11.40
s420	81.06	2290.24	19.40	81.58	2243.32	18.62
s444	88.10	2043.40	12.14	88.46	1999.20	12.14
s510	138.73	3442.84	15.22	138.90	3344.92	15.50
s641	86.93	2512.60	19.69	88.56	2372.52	20.12
s713	87.99	2513.28	21.85	90.18	2337.84	20.10
s820	139.58	4162.28	10.51	141.55	4009.28	10.67
s832	134.25	4223.48	11.33	138.28	4024.92	10.86
ttt2	121.67	2887.28	12.50	122.56	2748.56	12.91
x1	159.14	4293.52	8.13	161.59	4125.56	8.94
x3	406.71	10691.64	14.23	409.77	10643.02	12.73
x4	191.04	4870.84	15.75	190.67	4784.48	15.26

Table 5: Technology decomposition and mapping results C : *pd_map* with *min-power_t_decomp*, D :*pd_map* with *bh_minpower_t_decomp*.

Figure 7: Number of nets vs. switching rate for s832 using *ad_map* versus *pd_map*.

Figure 8: Average load per net vs. switching rate for s832 using *ad_map* versus *pd_map*.

6 Discussions and Extensions

We assumed a zero-delay model (which does not account for power consumption due to glitches) during the technology mapping phase. This shortcoming can be overcome by using a real delay model (such as the SIS pin-dependent library delay model) when calculating the expected number of transitions. Using a real delay model has however several drawbacks. The first is the huge computational effort to calculate the possible glitches for each nodes. In [9], symbolic simulation is used to produce a set of Boolean functions which represent conditions for switching at each gate in the circuit at a specific time instance. This procedure is exact, however, requires large computation time and storage space. In [20], a faster method of estimating the power consumption including glitches is described using the notion of transition probability. The transition probabilities are propagated from the primary inputs to the circuit outputs using a linear time algorithm. Although the later method significantly reduces the computation time and space requirement, it is still costly.

The second drawback is that under a real delay model, the dynamic programming based tree mapping algorithm does **not** guarantee to find an optimum solution even for a tree. The mapping algorithm assumes that the current best solution is derived from the best solutions stored at the fanin nodes of the matching gate. This is true for power estimation under a zero delay model, but not for that under a real delay model. Consider Figure 9 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 (Figure 9a). 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 (Figure 9b). So if S_1 is dropped from the power-delay curve, the optimal solution may not be found. The reason is that glitches depend on the gate delay and the transition waveforms of the gate inputs which are not related to the minimum power mapping solutions for the inputs. It is very expensive to store all partial solutions, so we have to drop the locally inferior solutions. The dynamic programming approach only acts as a heuristic approach (even for trees).

Another assumption we made was that primary inputs are pairwise uncorrelated. If this assumption is relaxed, we must use the conditional probabilities to calculate the signal probability at the outputs of intermediate nodes. It is, however, too costly to consider conditional probabilities among all subsets of primary inputs. Instead, we only use pairwise conditional probabilities as follows. Correlation coefficient $C(i,j)$ is defined by

$$P(ij) = P(i)P(j|i) = P(i)P(j)C(i,j), \tag{42}$$

Figure 9: An example to illustrate complications due to a real delay model.

that is,

$$C(i,j) = \frac{P(i|j)}{P(i)} = \frac{P(j|i)}{P(j)}. \tag{43}$$

Ercolani et al. [8] describe a method to approximate the correlation coefficients of the outputs of gates given the signal probabilities and correlation coefficients of the inputs as follows. Let g be a gate with inputs i and j and the correlation coefficients of i and j, i and m, and j and m be given as $C(i,j)$, $C(i,m)$ and $C(j,m)$. The correlation coefficient of g and m is approximated by
$g =$ AND gate:

$$C(g,m) = C(i,m)C(j,m), \tag{44}$$

$g =$ OR gate:

$$C(g,m) = \frac{P(i)C(i,m) + P(j)C(j,m) - P(i)P(j)C(i,m)C(j,m)C(i,j)}{P(i) + P(j) - P(i)P(j)C(i,j)}, \tag{45}$$

$g =$ NOT gate:

$$C(g,m) = \frac{1 - P(i)C(i,m)}{1 - P(i)}. \tag{46}$$

The signal probability of a product term is estimated by breaking down the implicant into a tree of 2-input AND gates and then using the above formula to calculate the correlation coefficients of the internal nodes and hence the signal probability at the output. Similarly, the signal probability of a sum term is estimated by breaking down the implicate into a tree of 2-input OR gates.

Several extensions can be made on the *pd-map* to improve its effectiveness and versatility. First, pin permutation can be considered during the gate matching procedure. In general, library gates have pins that are functionally equivalent which means that inputs can be permuted on those pins without changing function of the gate output. These equivalent pins may have different input pin loads and pin dependent delays. If high switching activity inputs are matched with pins that have low input load, the power consumption can be reduced. In particular, the internal power consumption depends on the switching activities and the pin assignment of the input signals. To find the minimum power pin assignment for gate g matching at node n, we must solve the following optimization problem:

$$\min_{\pi \in PP(n,g)} \sum_{i \in pins(g)} C_G^i E(\pi(i)) + \sum_{j \in internal\_nodes(g)} C_{SD}^j E_j(charge/discharge, \pi) \tag{47}$$

where $PP(n,g)$ is the set of all valid pin permutations for gate g matching at node n, $pins(g)$ is the set of pins of g, $\pi(i)$ is the input that is mapped to pin i under pin

permutation π, and $E_j(charge/discharg, \pi)$ is the expected number of charge/discharge events for an internal node j under pin permutation π.

Taking the 3-input NOR gate in Figure 3 as an example, the power cost for the specified pin permutation, assuming inputs are uncorrelated, is calculated as

$$\begin{aligned} P_{avg} &= P(i_1)\,[1-P(i_1)]\,C_G^{p_1} + P(i_2)\,[1-P(i_2)]\,C_G^{p_2} + P(i_3)\,[1-P(i_3)]\,C_G^{p_3} \\ &+ \frac{[1-P(i_1)]\,[P(i_1)(1-P(i_2))(1-P(i_3))]}{[1-P(i_1)] + [P(i_1)(1-P(i_2))(1-P(i_3))]} C_{SD}^{A} \\ &+ \frac{[1-P(i_1)]\,[1-P(i_2))((1-P(i_3))(P(i_1)+P(i_2)-P(i_1)P(i_2))]}{[1-P(i_1)]\,[1-P(i_2)) + ((1-P(i_3))(P(i_1)+P(i_2)-P(i_1)P(i_2))]} C_{SD}^{B} \\ &+ [(1-P(i_1))(1-P(i_2))(1-P(i_3))]\,[1-(1-P(i_1))(1-P(i_2))(1-P(i_3))]\,C_{SD}^{out}. \end{aligned}$$

The optimum solution can be found by enumerating all valid pin permutations. This is not so costly since the number of pins for typical library gates is small (< 6).

The pin permutation can be directly incorporated in the technology mapping process as follows. For each match g, all equivalent pin permutations are generated and the corresponding power and delay costs are calculated. Permutations that result in inferior *power-delay* trade-off points will be dropped. Permutations that result in non-inferior solutions are stored in the power delay trade-off curve along with the corresponding power and delay values. This may lead to a large number of points on the trade-off curve. In that case, we can restrict the number of points on the curves to a user defined number M and thus manage the space complexity (see Section 4.5).

The concept of power delay trade-off curve can be extended to include the area trade-off. Instead of generating a set of (power, delay) values, the trade-off curve can consist of a set of (power, delay, area) values. One drawback of the three dimensional trade-off space is that the number of points which must be generated in order to find good mapping solutions, is very large. A balance between optimality and computation cost has to be reached in order to determine how many tradeoff points should be stored.

7 Concluding Remarks

We first described an extended power consumption model which includes the power consumption at the internal nodes of a gate. We then showed how to calculate the internal power consumption based on the notion of charging and discharging probabilities and transistor graphs. Experimental results show that the internal power consumption can be as much as 16.5% of the power consumption due to gate capacitances and hence is significant. Based on the extended power consumption model, we presented algorithms for

low power technology decomposition and mapping. In particular, we generate networks with minimum total switching activity and feed them to a delay constrained power efficient technology mapper which *hides* the highly active nodes inside the mapped gates. Experimental results show that significant reduction in power consumption is achieved without performance degradation.

Technology mapping for low power under a real delay model was also addressed where we showed that even for trees, optimal mapping solution cannot be obtained. We then described a method to estimate the average power consumption when the primary inputs are correlated. Extensions to the power efficient technology mapper were also presented.

References

[1] MOSIS data sheet, Information Sciences Institute, University of Southern California,. 1992.

[2] R. K. Brayton, G. D. Hachtel, and A. L. Sangiovanni-Vincentelli. Multilevel logic synthesis. In *Proceedings of the IEEE*, volume 78, pages 264–300, February 1990.

[3] R. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, C-35:677–691, Aug. 1986.

[4] A. R. Burch, F. Najm, P. Yang, and D. Hocevar. Pattern independent current estimation for reliability analysis of CMOS circuits. In *Proceedings of the 25th Design Automation Conference*, pages 294–299, June 1988.

[5] 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.

[6] M. A. Cirit. Estimating dynamic power consumption of CMOS circuits. In *Proceedings of the IEEE International Conference on Computer Aided Design*, pages 534–537, November 1987.

[7] E. Detjens, G. Gannot, R. Rudell, A. Sangiovanni-Vincentelli, and A. Wang. Technology mapping in MIS. In *Proceedings of the IEEE International Conference on Computer Aided Design*, pages 116–119, November 1987.

[8] S. Ercolani, M. Favalli, M. Damiani, P. Olivo, and B. Ricco. Estimate of signal probability in combinational logic networks. In *Proceedings of the European Test Conference*, pages 132–138, 1989.

[9] 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.

[10] D. A. Huffman. A method for the construction of minimum redundancy codes. In *Proceedings of the IRE*, volume 40, pages 1098–1101, September 1952.

[11] D. S. Parker Jr. Conditions for optimality of the Huffman algorithm. *SIAM Journal of Computing*, 9(3):470–489, August 1980.

[12] K. Keutzer. Dagon: technology binding and local optimization by dag matching. In *Proceedings of the 24th Design Automation Conference*, pages 341–347, June 1987.

[13] L. Larmore and D. S. Hirschberg. A fast algorithm for optimal length-limited Huffman codes. *Journal of the Association for Computing Machinery*, V 37 No. 3:464–473, 1990.

[14] 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.

[15] 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.

[16] 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.

[17] K. Roy and S. Prasad. Syclop: Synthesis of CMOS logic for low power application. In *Proceedings of the International Conference on Computer Design*, pages 464–467, October 1992.

[18] H. Savoj and H. Y. Wang. Improved scripts in MIS-II for logic minimization of combinational circuits. In *Proceedings of the International Workshop on Logic Synthesis*, May 1991.

[19] 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.

[20] C. Y. Tsui, M. Pedram, and A. Despain. Efficient estimation of dynamic power dissipation with an application. In *Proceedings of the IEEE International Conference on Computer Aided Design*, Nov 1993. to appear.

[21] H. Vaishnav and M. Pedram. Pcube: A performance driven placement algorithm for low power designs. In *Proceedings of the EURO-DAC*, September 1993. to appear.

[22] H. J. M. Veendrick. Short-circuit dissipation of static CMOS circuitry and its impact on the design of buffer circuits,. IEEE *Journal of Solid State Circuits*, SC-19:468–473, Aug. 1984.