# Power Estimation and Optimization at the Logic Level

Massoud Pedram
Department of EE - Systems
University of Southern California
Los Angeles, CA 90089

# Contents

# Abstract

This paper describes various approaches for power analysis and minimization at the logic level including, amongst others, pattern-independent probabilistic and symbolic simulation techniques for power estimation and low power techniques for state assignment, logic restructuring, logic decomposition, technology mapping and pin ordering.

# 1 Introduction

Low power, yet high-throughput and computationally intensive, circuits are becoming a critical application domain. One driving factor behind this trend is the growing class of personal computing devices (digital pens, portable desktops, audio- and video-based multimedia products) as well as wireless communications and imaging systems (personal digital assistants, personal communicators, smart cards) which demand high-speed computations, complex functionalities and often real-time processing capabilities with low power consumption. Another crucial driving factor is that excessive power consumption is becoming the limiting factor in integrating more transistors on a single chip or on a multiple-chip module. Unless power consumption is dramatically reduced, the resulting heat will limit the feasible packing and performance of VLSI circuits and systems. Indeed, circuits synthesized for low power are also less susceptible to run time failures. Exploring the tradeoff between area, performance and power during synthesis and design is thus demanding more attention.

Low power VLSI design can be achieved at various levels. For example, at the algorithmic level, correct data representation and choice of algorithms may significantly reduce power consumption [11]. At the system level, inactive hardware modules may be automatically turned off to save power; modules may be provided with the optimum supply voltage and interfaced by means of level converters. At the architectural design level, concurrency increasing transformations such as loop unrolling, pipelining and control flow optimization as well as critical path reducing transformations such as height minimization, retiming and pipelining may be used to allow a reduction in supply voltage without degrading system throughput [13, 12]; Algorithm-specific instruction sets may be utilized that boost code density and minimize switching; A Gray code addressing scheme can be used to reduce the number of bit changes on the address bus [60]; Internal busses may be replaced by point-to-point connections to avoid driving a large number of modules on every bus access; Bus architectures with sub 1-volt swing and low standby current can be used [44]; On-chip cache may be added to minimize external memory references; Locality of reference may be exploited to avoid accessing global resources such as memories, busses or ALU's; Control signals that are "don't cares" can be held constant to avoid initiating nonproductive switching. At the device level, threshold voltage of MOS transistors can be reduced to match the reduced supply voltage [33, 58]; Very low threshold voltages may be made possible by electrically

controlling the threshold values (against process and temperature variations) by substrate modulation [9].

Logic synthesis fits between the register transfer level specification and the netlist of gates specification. It provides the automatic synthesis of netlists minimizing some objective function subject to various constraints. Example inputs to a logic synthesis system include two-level logic representation, multi-level Boolean networks, finite state machines and technology mapped circuits. Depending on the input specification (combinational versus sequential, synchronous versus asynchronous), the target implementation (two-level versus multi-level, unmapped versus mapped, ASICs versus FPGAs), the objective function (area, delay, power, testability) and the delay models used (zero-delay, unit-delay, unit-fanout delay, or library delay models), different techniques are applied to transform and optimize the original RTL description.

Once the system level, architectural and technological choices are made, it is the switching activity of the logic (weighted by the capacitive loading) that determines the power consumption of a circuit. In this paper, we will describe a number of techniques for power estimation and minimization at the logic level. Our emphasis during power estimation will be on pattern-independent simulation techniques while our strategy for synthesizing circuits for low power consumption will be to restructure/optimize the circuit to obtain low switching activity factors at nodes which drive large capacitive loads.

The remainder of the paper is organized as follows. In Section 2, we review various power estimation techniques including those that assume a zero (non-glitch) delay model and those that assume a real delay model which accounts for glitches and hazards. Issues related to spatial and temporal correlations among circuit inputs will be addressed as well. In Section 3, we describe some techniques for combinational and sequential logic optimization targeting low power consumption including multi-level network optimization, common subexpression extraction, technology decomposition, technology mapping and signal-to-pin assignment. Concluding remarks are given in Section 4.

# 2 Power Estimation Techniques

## 2.1 Sources of Power Dissipation

Power dissipation in CMOS circuits is caused by three sources: the (subthreshold) leakage current which arise from the inversion charge that exists at the gate voltages below the conventional threshold voltage [39], the short-circuit current [67] which is due to the DC path between the supply rails during output transitions, and the charging and discharging of capacitive loads during logic changes.

The subthreshold current for long channel devices increases linearly with the ratio of the channel width over channel length and decreases nearly exponentially with decreasing $V_{GT} = V_{GS} - V_T$ where $V_{GS}$ is the gate bias and $V_T$ is the threshold voltage. Several hundred millivolts of "off bias" (say, 300-400 $mV$) typically reduces the subthreshold current to negligible values. With reduced power supply and device threshold voltages, the subthreshold currents will however become more pronounced. In addition, at short channel lengths, the subthreshold current also becomes exponentially dependent on drain voltage instead of being independent of $V_{DS}$ (see [19] for a recent analysis).

[67] shows that the short-circuit power consumption for an inverter gate is proportional to gain of the inverter, the cubic power of supply voltage minus device threshold, the input rise/fall time, and the operating frequency. The maximum short circuit current flows when there is no load; this current decreases with the load. If gate sizes are selected so that the input and output rise/fall times are about equal, the short-circuit power consumption will be less than 15% of the dynamic power consumption. If, however, design for high performance is taken to the extreme where large gates are used to drive relatively small loads, then there will be a stiff penalty in terms of short-circuit power consumption.

It is generally agreed that the short-circuit and subthreshold currents in CMOS circuits can be made small with proper circuit and device design techniques. The dominant source of power dissipation is thus the charging and discharging of the node capacitances (also referred to as the dynamic power dissipation) and is given by:

$$P_{avg}(g) = \frac{1}{2T_{cycle}} \left( V_{dd}^2 C_g E_g(sw) + \sum_{i \in internal\_nodes(g)} V_{dd}(V_{dd} - V_T) C_i^{SD} E_i(c/d) \right)$$

where $T_{cycle}$ is the clock cycle time, $V_{dd}$ is the supply voltage,

$$C_g = C^{wire} + C_g^{SD} + \sum_{j \in fanout(g)} C_j^G,$$

$C^{wire}$ is the wiring capacitance of the net driven by $g$, $C_x^{SD}$ is the source-drain diffusion capacitance at node $x$, $C_j^G$ is the gate capacitance of $j$, $E_g(sw)$ is the expected number of transitions at the output of $g$ per clock cycle, $V_T$ is the threshold voltage of the device ($V_T = V_{T_n} = -V_{T_p}$), and $E_i(c/d)$ is the expected number of charge and discharge events at $i$ per clock cycle. Note that an internal node in the $n$-subnetwork of a CMOS gate never reaches a voltage above $V_{dd} - V_T$ through the normal pull-up mechanism. Similarly, no node in the $p$-subnetwork reaches a voltage below $V_T$ through the normal pull-down mechanism.

Calculation of $E(sw)$ is difficult as it depends on (1) the input patterns and the sequence in which they are applied, (2) the delay model used and (3) the circuit structure.

To appreciate the effect of various correlations on $E(sw)$ consider a two-input AND gate $g$ with independent inputs $i$ and $j$ whose signal probabilities are 1/2, then $E_g(sw) = 3/8$. Now suppose it is known that only patterns 00 and 11 can be applied to the gate inputs and that both patterns are equally likely, then $E_g(sw) = 1/2$. Alternatively, assume that it is known that every 0 applied to input $i$ is immediately followed by a 1 while every 1 applied to input $j$ is immediately followed by a 0, then $E_g(sw) = 4/9$. The first case is an example of spatial correlations between gate inputs while the second case illustrates temporal correlations on spatially independent gate inputs.

Based on the delay model used, the power estimation techniques could account for steady-state transitions (which consume power, but are necessary to perform a computational task) and/or hazards and glitches (which dissipate power without doing any useful computation). It has been shown in [2] that although the mean value of the ratio of hazardous component to the total power dissipation varies significantly with the considered circuits (from 9% to 38%), the hazard/glitch power dissipation cannot be neglected in static CMOS circuits. Indeed, an average of 15-20% of the total power is dissipated in glitching. The glitch power problem is likely to become even more important in future scaled technology.

In real networks, statistical perturbations of circuit parameters may change the propagation delays and produce changes in the number of transitions because of the appearance or disappearance of hazards. It is therefore useful to determine the change in the signal transition count as a function of this statistical perturbations. Variation of gate delay pa-

rameters may change the number of hazards occurring during a transition as well as their duration. For this reason, it is expected that the hazardous component of power dissipation is more sensitive to IC parameter fluctuations than the power strictly required to perform the transition between the initial and final state of each node [2].

The major difficulty in computing the signal probabilities is the reconvergent nodes. Indeed, if a network consists of simple gates and has no reconvergent fanout stems (or nodes), then the exact signal probabilities can be computed during a single post-order traversal of the network. For networks with reconvergent fanout, the problem is much more difficult.

In a typical 0.8 $\mu$m technology, the source/drain capacitance is about 20% of the gate capacitance [23]. 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. This is especially important as the internal capacitances may be charged and discharged without any change in the gate output. Internal power consumption is more manifest for complex gates which have more internal nodes with higher parasitic capacitances.

Calculation of $E_i(c/d)$ is even more difficult as an internal node may be floating (neither connected to $V_{dd}$ nor $Gnd$). In [63], it is shown that, under a zero delay model, this problem reduces to that of calculating the high and low signal probabilities at $i$ through the following equation:

$$E_i(c/d) = \frac{prob_{high}(i)prob_{low}(i)}{prob_{high}(i) + prob_{low}(i)}$$

where $prob_{high}(i)$ and $prob_{low}(i)$ denote probabilities of $i$ being charged to $V_{dd}$ and discharged to $Gnd$ and are calculated from the transistor graph of gate $g$. In this graph, each node represents an internal node of the gate, $V_{dd}$ and $Gnd$ and each edge represents a transistor of the gate. By doing a depth first traversal on the transistor graph, one can find all paths connecting $n$ to the $V_{dd}$ and $Gnd$, $prob_{high}(n)$ and $prob_{low}(n)$ can be easily computed. For a set of benchmark circuits, experimental results indicate that power consumption at the internal nodes is about 16% of the power consumption at the external nodes.

## 2.2 Switch-Level Simulation

Circuit simulation based techniques [41, 25, 66, 46] simulate the circuit with a representative set of input vectors. They are accurate and capable of handling various device models, different circuit design styles, dynamic / precharged logic tristate drives, latches, flip-flops,

etc. Although circuit level simulators are accurate, flexible and easy-to-use, they suffer from memory and execution time constraints and are not suitable for large, cell-based designs. In general, it is difficult to generate a compact stimulus vector set to calculate accurate activity factors at the circuit nodes.

A Monte Carlo approach for power estimation which alleviates this problem has been proposed in [8]. The convergence time for this approach is quite good when estimating the total power consumption of the circuit. However, when signal probability (or power consumption) values on individual lines of the circuit are required, the convergence rate is not so good.

Switch-level simulation techniques [6, 47, 51] are in general orders of magnitude faster than circuit-level simulation techniques, but are not as accurate or versatile. [14] describes *PowerMill*, a transistor-level power simulator and analyzer which estimates the dynamic power dissipation (including that due to hazards and glitches), detects dc leakage path and estimates short-circuit transient currents.

## 2.3 Estimation for Combinational Circuits

### 2.3.1 Estimation under a Zero Delay Model

Most of the power in CMOS circuits is consumed during charging and discharging of the load capacitance. To estimate the power consumption, one has to calculate the (switching) activity factors of the internal nodes of the circuit. Methods of estimating the activity factor $E_n(sw)$ at a circuit node $n$ involve estimation of signal probability $prob(n)$, which is the probability that the signal value at the node is one. Under the assumption that the values applied to each circuit input are temporally independent, we can write:

$$E_n(sw) = 2prob(n)(1 - prob(n)).$$

Computing signal probabilities has attracted much attention. [48] presents some of the earliest work in computing the signal probabilities in a combinational network. The authors associate variable names with each of the circuit inputs representing the signal probabilities of these inputs. Then, for each internal circuit line, they compute algebraic expressions involving these variables. These expressions represent the signal probabilities for these lines. While the algorithm is simple and general, its worse case time complexity is exponential.

Two distinct (directed) paths are reconvergent if they start at a common node (say $A$) and terminate at another common node (say $B$). These paths are said to *fanout* at $A$ and *reconverge* at $B$. $A$ and $B$ are refered to as the reconvergent fanout stem and the reconvergent node, respectively. The major difficulty in computing the signal probabilities is the reconvergent nodes. Indeed, if a network consists of simple gates and has no reconvergent fanout stems (or nodes), then the exact signal probabilities can be computed during a single post-order traversal of the network using the following equations [21]:

$$\begin{aligned}
&\text{NOT gate:} && prob(o) = 1 - prob(i), \\
&\text{AND gate:} && prob(o) = \textstyle\prod_{i \in inputs} prob(i), \\
&\text{OR gate:} && prob(o) = 1 - \textstyle\prod_{i \in inputs} (1 - prob(i)).
\end{aligned}$$

This simple algorithm is known as the *tree algorithm* as it produces the exact signal probabilities for a tree network. For networks with reconvergent fanout, the tree algorithm however yields approximate values for the signal probabilities.

In [56], a graph-based algorithm is used to compute the exact signal probabilities using Shannon's expansion. This algorithm relies on the notion of the supergate of a node $n$ which is defined as the smallest sub-circuit of the (transitive fanin) cone of $n$ whose inputs are independent, that is, the cones of the inputs are mutually disjoint. Maximal supergates are the supergates which are not properly contained in any other supergate. Maximal supergates form a unique cover of circuit. Although this cover is disjoint for a single-output circuit, it is not necessarily disjoint for a multiple-output circuit [55]. The procedure, called the *super-gate algorithm*, essentially requires three steps: 1) Identification of reconvergent nodes, 2) Determination of the maximal supergates, and 3) Calculation of signal probabilities at the supergate outputs based on the Shannon's expansion with respect to all multiple-fanout inputs of the supergate. In the worst case, a supergate may include all the circuit inputs as multiple-fanout inputs. In such cases, the super-gate algorithm becomes equivalent to an exhaustive true-value simulation of the supergate.

[28] provides an extension to the tree algorithm, called the *weighted averaging algorithm*. The new algorithm computes the signal probability at the output of a gate approximately using Shannon's expansion with respect to each multiple-fanout primary input in the support of the gate. The signal probability of the gate is then calculated as a weighted sum of these approximate signal probabilities. This approach attempts to take into account the first order

effects of reconvergent fanout stems in the variable support of the node. It is linear in the product of the number of circuit inputs and the size of the circuit.

[52] gives an algorithm, known as the *cutting algorithm*, which computes lower and upper bounds on the signal probability of reconvergent nodes by cutting the multiple-fanout reconvergent input lines and assigning an appropriate probability range to the cut lines and then propagating the bounds to all the other lines of the circuits by using propagation formulas for trees. The effectiveness of the cutting algorithm, however, depends on the non deterministic choice of the cuts. Well-chosen cuts lead to better estimates of the signal probabilities while poorly chosen cuts results in poor estimates. The algorithm runs in polynomial time in terms of the size of the circuits.

[42] gives an exact procedure based on Ordered Binary-Decision Diagrams (OBDDs) [5] which is linear in the size of the corresponding function graph (the size of the graph, of course, may be exponential in the number of circuit inputs). The signal probability at the output of a node is calculated by first building an OBDD corresponding to the global function of the node and then performing a postorder traversal of the OBDD using equation:

$$prob(y) = prob(x)prob(f_x) + prob(\bar{x})prob(f_{\bar{x}}).$$

[18] presents a procedure for propagating signal probabilities from the circuit inputs toward the circuit outputs using only pairwise correlations between circuit lines and ignoring higher order correlation terms as follows. The correlation coefficient of $i$ and $j$ is defined as:

$$C(i,j) = \frac{prob(i \wedge j)}{prob(i)prob(j)}.$$

Ignoring higher order correlation coefficients, it is assumed that $C(i,j,k) = C(i,j)C(i,k)C(j,k)$. 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 signal probability of $g$ and the correlation coefficient of $g$ and $m$ are thus approximated by:

| | |
|---|---|
| NOT gate: | $prob(g) = 1 - prob(i)$, |
| | $C(g,m) = \frac{1-prob(i)C(i,m)}{1-prob(i)}$, |
| Two-input AND gate: | $prob(g) = prob(i)prob(j)C(i,j)$, |
| | $C(g,m) = C(i,m)C(j,m)$, |
| Two-input OR gate: | $prob(g) = prob(i) + prob(j) - prob(i)prob(j)C(i,j)$, |
| | $C(g,m) = \frac{prob(i)C(i,m)+prob(j)C(j,m)-prob(i)prob(j)C(i,m)C(j,m)C(i,j)}{prob(i)+prob(j)-prob(i)prob(j)C(i,j)}$. |

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.

[54, 34] model the temporal correlation between values of some signal $x$ in two successive clock cycles by a time-homogeneous Markov chain which has two states 0 and 1 and four edges where each edge $ij$ $(i, j = 0, 1)$ is annotated with the conditional probability $prob^x_{ij}$ that $x$ will go to state $j$ at time $t + 1$ if it is in state $i$ at time $t$. The transition probability $prob(x_{i\to j})$ is equal to $prob(x = i)prob^x_{ij}$. Obviously, $prob^x_{00} + prob^x_{01} = prob^x_{10} + prob^x_{11} = 1$ while $prob(x) = prob(x_{0\to 1}) + prob(x_{1\to 1})$ and $prob(\bar{x}) = prob(x_{0\to 0}) + prob(x_{1\to 0})$. The activity factor of line $x$ can be expressed in terms of these transition probabilities as follows:

$$E_x(sw) = prob(x_{0\to 1}) + prob(x_{1\to 0}).$$

The various transition probabilities can be computed exactly using the OBDD representation of the logic function of $x$ in terms of the circuit inputs.

[34] goes on to describe a mechanism for propagating the transition probabilities through the circuit which is more efficient as there is no need to build the global function of each node in terms of the circuit inputs, but is less accurate. The loss is often small while the computational saving is significant. It then extends the model to account for spatio-temporal correlations (i.e., spatial correlations between temporally-dependent events). The mathematical foundation of this extension is a four state time-homogeneous Markov chain where each state represents some assignment of binary values to two lines $x$ and $y$ and each edge describes the conditional probability for going from one state to next.

These methods only account for steady-state behavior of the circuit and thus ignore hazards and glitches. The next section reviews some techniques that examine the dynamic behavior of the circuit and can thus calculate the power dissipation due to hazards and glitches.

### 2.3.2 Estimation under a Real Delay Model

In [20], the exact power estimation of a given combinational logic circuit is carried out by creating a set of symbolic functions such that summing the signal probabilities of the functions

corresponds to the average switching activity at a circuit line $x$ in the original combinational circuit. The inputs to the created symbolic functions are the circuit input lines at time instances $0^-$ and $\infty$. Each function is the EXCLUSIVE OR of the characteristic functions describing the logic values of $x$ at two consecutive instances. The major disadvantage of this estimation method is its exponential complexity. However, for the circuits that this method is applicable to, the estimates provided by the method can serve as a basis for comparison among different approximation schemes.

In [20], the underlying delay model was the Fixed Binary Pure Delay Model. The disadvantage of this particular delay model is that it ignores static (process-dependent) and dynamic (slope factor, switching noise, temperature) variations in delay values. An approach using the Extended Bounded Delay (XBD) Model that can take those factors into consideration seems more appropriate. In the XBD model, the gate delays are represented by a range of values. The output of the gate during a period equal to its delay is assumed to be unknown or $X$.

In [16], a symbolic simulation technique for timing analysis under the XBD model is presented. The main idea is to construct a symbolic formula (in terms of binary values of the input vectors) which is a tautology exactly if the circuit outputs stabilize after some specified time instance. Power analysis is however more complex as one must compute the number of transitions at each node rather than simply the time at which the circuit outputs stabilize to binary values. In [17], it is shown that (under a reasonable assumption regarding the power dissipation of glitches and by interpreting $X$ as the slope factor), this problem can be transformed to the problem of signal probability calculation on a set of Boolean (characteristic) functions defined at specific time instances. Moreover, the logical waveforms produced under the XBD model satisfy the *block-monotone property*, i.e., the time axis for each waveform can be divided into a small number of blocks such that signal probabilities in any block are monotone increasing (or alternatively, monotone decreasing). This property is exploited to eliminate unnecessary calculation of signal probabilities, and thus significantly speed-up the symbolic simulation.

[7] introduces the concept of a *probability waveform* which consists of a sequence of transition edges or events over time from the initial steady state (time $0^-$) to the final steady state (time $\infty$) where each event is annotated with an occurrence probability. The probability waveform of a node is a compact representation of the set of all possible logical

waveforms at that node. Given these waveforms, it is straight-forward to calculate the switching activity of $x$ which includes the contribution of hazards and glitches, i.e.:

$$E_x(sw) = \sum_{t \in event\_list(x)} \left( prob(x^t_{0\to 1}) + prob(x^t_{1\to 0}) \right).$$

Given such waveforms at the circuit inputs and with some convenient partitioning of the circuit, the authors examine every sub-circuit and derive the corresponding waveforms at the internal circuit nodes. [43] describes an efficient technique to propagate transition waveforms at the circuit primary inputs up in the circuit and thus estimate the total power consumption (ignoring signal correlations due to reconvergent fanout).

Note that a probability waveform accounts for the temporal dependence in a limited way: It keeps track of the dependence between two signal values separated by a single transition edge, it however does not keep track of the dependence between signal values separated by more than one transition edge. Other methods to account for these correlations more exactly are required.

[59] improves the probabilistic simulation approach by calculating the statistics of the waveforms and delays more accurately and by considering the signal correlations using the method of [18]. [62] describes an efficient tagged probabilistic simulation approach that correctly accounts for reconvergent fanout and glitches. The key idea is to break the set of possible logical waveforms at a node $n$ into four groups, each group being characterized by its steady state values (i.e., values at time instance $0^-$ and $\infty$). Next, each group is combined into a probability waveform with the appropriate steady-state tag. Given the tagged probability waveforms at the input of a simple gate, it is then possible to compute the tagged probability waveforms at the output of the gate. The correlation between probability waveforms at the inputs is approximated by the correlation between the steady state values of these lines. This approach requires significantly less memory and runs much faster than symbolic simulation, yet achieves very high accuracy, e.g., the average error in aggregate power consumption is about 2%.

## 2.4 Estimation for Sequential Circuits

Recently developed methods for power estimation have primarily focused on combinational logic. The estimates produced by purely combinational methods can greatly differ from those

produced by the exact state probability method. Accurate average switching activity estimation for sequential circuits is considerably more difficult than for combinational circuits, because the probability of the circuit being in each of its possible states has to be calculated.

A first attempt at estimating switching activity in logic-level sequential circuits has been presented in [20]. This method can accurately model the correlation between the applied vector pairs, but assumes that the state probabilities are uniform. [65, 37] present results obtained by using the Chapman-Kolmogorov equations for discrete-time Markov Chains to compute the exact state probabilities of the machine. We describe the method below.

For each state $S_i,\ 1 \leq i \leq K$ in the STG, we associate a variable $prob(S_i)$ corresponding to the steady-state probability of the machine being in state $S_i$ at $t = \infty$. For each edge $e_{ij}$ in the STG, we have $I_{ij}$ signifying the input combination corresponding to the edge. Given static probabilities for the primary inputs to the machine, we can compute $prob(S_j|S_i)$, the conditional probability of going from $S_i$ to $S_j$. For each state $S_j$, we can write an equation:

$$prob(S_j) = \sum_{S_i \in in\_state(S_j)} prob(S_i) prob(S_j|S_i)$$

where $in\_state(S_i)$ is the set of fanin states of $S_i$ in the STG. Given $K$ states, we obtain $K$ equations out of which any one equation can be derived from the remaining $K-1$ equations. We have a final equation:

$$\sum_j prob(S_j) = 1.$$

This linear set of $K$ equations is solved to obtain the different $prob(S_j)$'s.

The Chapman-Kolmogorov method requires the solution of a linear system of equations of size $2^N$, where $N$ is the number of flip-flops in the machine. Thus, this method is limited to circuits with $\leq 15$ flip-flops, since it requires the explicit consideration of each state in the circuit.

[65, 37] go on to describe a framework for approximate switching activity estimation of sequential circuits. The basic computation step is the solution of a non-linear system of equations. In the following, we derive the system of non-linear equations which is solved when calculating the state bit probabilities. We can write:

$$\begin{aligned} \mathcal{N}_1 &= \mathcal{F}_1(PI, \mathcal{P}_1, \mathcal{P}_2, \ldots) \\ &\vdots \\ \mathcal{N}_n &= \mathcal{F}_n(PI, \mathcal{P}_1, \mathcal{P}_2, \ldots) \end{aligned}$$

where $\mathcal{N}_i$ and $\mathcal{P}_j$ denote the $i^{th}$ next state bit at the output and the $j^{th}$ present state bit at the input of a finite state machine, respectively and $\mathcal{F}_l$'s are the next-state (Boolean) functions. Alternatively, we can write:

$$\begin{aligned} ns_1 &= f_1(pi, ps_1, ps_2, \ldots, ps_n) \\ &\vdots \\ ns_n &= f_n(pi, ps_1, ps_2, \ldots, ps_n) \end{aligned}$$

where $ns_i$ and $ps_j$ denote the state bit probabilities of the $i^{th}$ next state bit at the output and the $j^{th}$ present state bit at the input of the FSM, respectively and $f_l$'s are nonlinear algebraic functions. Since we want to find the steady state bit probabilities, hence the above system becomes:

$$\begin{aligned} ps_1 &= f_1(pi, ps_1, ps_2, \ldots, ps_n) \\ &\vdots \\ ps_n &= f_n(pi, ps_1, ps_2, \ldots, ps_n) \end{aligned}$$

which is a system of non-linear equations. The fixed point (or zero) of this system of equations can be found using the Picard-Peano (or Newton-Raphson) iteration [29].

Increasing the number of variables or the number of equations in the above system results in increased accuracy [38]. For a wide variety of examples, it is shown that the approximation scheme is within 1-3% of the exact method, but is orders of magnitude faster for large circuits. Previous sequential switching activity estimation methods can have significantly greater inaccuracies.

# 3 Logic Optimization Techniques

## 3.1 State Assignment

It is well-known that the state assignment of a finite state machine (FSM) has a significant impact on the area, delay and power consumption of its final logic implementation. In the past, many researchers have addressed the encoding problem for minimum area of two-level [36, 68, 69] or multi-level logic [15, 31, 32] implementations. [50] describes a state assignment

procedure which minimizes the following cost function:

$$\sum_{i,j} W_{ij} H(S_i, S_j)$$

where $W_{ij}$ reflects the state transition probability from state $S_i$ to state $S_j$ and $H(S_i, S_j)$ is the Hamming distance between the encodings of the two states. The authors incorrectly use the conditional probabilities $prob(S_j|S_i)$ (rather than the state transition probabilities $prob(S_i \rightarrow S_j)$) for $W_{ij}$. The state transition probabilities are calculated prior to state assignment by solving the Chapman-Kolmogoroph equations to obtain the state probabilities and then multiplying the conditional probabilities by the appropriate state probabilities, that is, $prob(S_i \rightarrow S_j) = prob(S_i)prob(S_j|S_i))$ for $W_{ij}$.

This problem is equivalent to embedding a weighted graph on a hypercube of minimum (or given) dimensionality which is an NP-hard problem. It is therefore solved using simulated annealing [27]. The shortcomings of the above approach are: (1) It minimizes the switching on the present state bits without any consideration for the loading on the state bits; (2) It does not account for the power consumption in the resulting two- or multi-level logic realization of the next state logic of the FSM. In an attempt to account for power consumption in the combinational part of the FSM, [45] minimizes a linear combination of the number of state bits that change every cycle and the number of literals in a multi-level logic implementation of the FSM using a genetic local search algorithm. In the following, we describe a state assignment technique that not only accounts for the power consumption at the state bit lines, but also the power consumption in the combinational logic that implements the next state logic function.

The conventional approach to state-assignment for two-level logic is to assign one-hot codes to the states and then generate a minimum symbolic (multi-valued) cover of the machine by output-disjoint minimization. This symbolic cover defines a set of *face embedding constraints* which require that certain states be given codes that lie on the same face of a hypercube of minimum (or given) dimensionality. If these constraints are satisfied, then the number of cubes in the minimum binary cover of the machine will be upper bounded by that in the symbolic cover. The input encoding problem is to find a minimum encoding length such that all face constraints are satisfied while the bit-constrained encoding problem is to find an encoding that satisfies as many constraints as possible given the encoding length. In [36], the input encoding problem is solved heuristically using a row encoding technique.

In [69], the authors transform the input encoding problem into a unate covering problem (covering seed dichotomies by a minimum-cost set of prime dichotomies) and solve it using a heuristic technique. The dichotomy-based approach can be extended to solve the state assignment problem for low power. The main difference here is that the bit constrained encoding problem becomes a constrained rectangle covering problem which is NP-hard. [61] presents a heuristic solution to this rectangle covering problem.

The conventional approach to state assignment for multi-level logic is to use the state transition graph [15, 31] (or an optimized Boolean network obtained after one-hot encoding of the states [32]) to assign a weight to each pair of state symbols. This weight measures the desirability (or literal saving) of giving the two states codes that are as close as possible (or adjacent). The embedding algorithm identifies clusters of states that are joined by maximal weight edges and assigns to them minimally distant codes in a greedy fashion or by using simulated annealing. In the latter case, the literal cost of a given encoding is decreased by iteratively swapping codes of pairs of states. The work of [31] can be extended to target low power. In this case, the edge weights must reflect the power-saving achieved by coding the states joined by that edge a Hamming distance of one. A serious difficulty arises from the fact that the edge weights cannot be calculated independently as the power reduction due to grouping a pair of states varies as a function of encoding of the remaining states. One must therefore develop a procedure for dynamic calculation of the edge weights. A straight-forward approach is to start with an encoding that maximizes the literal saving and iteratively modify it to reduce the power consumption. [61] describes one such technique using simulated annealing.

## 3.2 Multi-Level Network Optimization

Network don't cares can be used for minimization of nodes in a boolean network [1]. Once the compatible don't cares [40] are computed for nodes in a network, each node can be optimized for area without any concern that changes in the function of this node might affect the function of primary outputs of the network. In [53] a procedure is introduced where observability don't cares and image projection techniques are used to compute the compatible local don't cares for each node in the network. (In this section, the term "local function" of a node refers to the function of node in terms of its immediate fanin in the

network while the term "global function" refers to the function of node in terms of the circuit inputs.) The compatible local don't care for node $n_i$ is then used to minimize the number of literals in the logic expression for $n_i$.

[57] presents a multi-level network optimization technique for low power. The difference between their procedure and the procedure in [53] is in the cost function used during the two-level logic minimization. Their cost function minimizes a linear combination of the number of product terms and the weighted switching activity. A major shortcoming of this approach is that it does not consider how changes in the global function of an internal node affects the signal probability (and thus, the switching activity) of nodes in its transitive fanout. In general, changing the global function of an internal node may change the signal probability of all nodes in its transitive fanout such that the increase in power consumption due to these fanout nodes may exceed any local power reduction.

It is therefore necessary to consider not only the local reduction in power consumption, but also the overall impact of the global function change on the switching activity of other nodes in the circuit. In this regard, two types of local don't care conditions can be identified: (1) *Function Preserving Don't Care* (FPDC) which consists of all points in the local space of the node which never occur and is simply the complement of the range of the primary input space into the space of the local fanins of the node and (2) *Function Modifying Don't Care* (FMDC) which consists of all points in the local space which do not produce observable outputs. FPDC does not affect the global function of the node while FMDC does. Consequently, when simplifying a node, one can freely use don't care conditions (DC-terms) from its FPDC, but should be cautious when using DC-terms from its FMDC as this may adversely increase the switching activity in the transitive fanout cone of the node.

[22] presents a greedy, yet effective, network optimization procedure as outlined below. The procedure proceeds in a reverse topological fashion from the circuit outputs to the circuit inputs simplifying fanouts of a node before reaching that node. Once a node $n$ is simplified, the procedure propagates those don't care conditions which could only increase (or decrease) the signal probability of that node if its current signal probability is greater than (less than or equal to) 0.5. This will ensure that once a node in the transitive fanin of $n$ is simplified, the switching activity of $n$ can only go down. Due to this don't care propagation strategy, all DC-terms in the node's FMDC can be used freely as they will not adversely impact the switching activity in the transitive fanout of the node.

## 3.3 Common Subexpression Extraction

Extraction based on algebraic division (using either kernels [4, 3] or two-literal cubes and double-cube divisors [49]) has proven to be very successful in creating an area-optimized multi-level Boolean network. [50] modifies the kernel extraction procedure to generate multi-level circuits with low power consumption as follows. Let $d = d(i_1, i_2, \cdots, i_K)$; $K \geq 1$, be a common subexpression of functions $f_1, f_2, \cdots, f_P$; $P \geq 2$. Let $j_1, j_2, \cdots, j_M$; $M \geq 0$ be the nodes internal to $d$.

When $d$ is factored out of $f_i$, the signal probabilities and switching activity at all nodes of the network remain unchanged. However, the loads at the output of the driver gates $i_1, \cdots, i_K$ change. Each gate now drives $P-1$ fewer gates. At the same time, since there is only one copy of $d$ instead of $P$ copies, there are $P-1$ fewer copies of internal nodes $j_1, j_2, \cdots, j_M$. The power saving in extracting $d$ is thus given by:

$$\frac{C_0 V_{dd}^2}{2T_{cycle}}(P-1)\left(\sum_{k=1}^{K} n_{i_k} E_{i_k}(sw) + \sum_{m=1}^{M} n_{j_m} E_{j_m}(sw)\right)$$

where $n_{i_k}$ is the number of gates belonging to $d$ and driven by signal $i_k$ and $n_{j_m}$ is the number of gates internal to $d$ and driven by signal $j_m$. This gain is correct assuming the load capacitance due to each fanout is equal to some constant value $C_0$ and that $f_i$'s are partially factorized (with respect to $d$) even before $d$ is extracted. The authors iteratively select and substitute a candidate divisor which minimizes a linear combination of literal-saving and power-saving factors.

In the following, we describe an alternate power-saving factor for double-cube divisors of [49] which assumes that nodes in the multi-level network are in two-level logic form. This is consistent with the assumption made for calculating the literal-saving cost of a candidate divisor. Consider a single-output Boolean function $f$ with cubes $c_1, c_2, \cdots, c_N$. Let $d = d_1 + d_2$ (where $d_1 = c_i/(c_i \bigcap c_j)$ and $d_2 = c_j/(c_i \bigcap c_j)$) be some double cube divisor which appears $P$ times in the expression for $f$, that is, $c_1 + c_2 = b_1 \bigcap d; c_3 + c_4 = b_2 \bigcap d; \cdots; c_{2P-1} + c_{2P} = b_P \bigcap d$ where $b_i$'s are the various bases corresponding to the $P$ copies of the divisor. After extraction, $d$ is made into a new node and $c_1, c_2, \cdots, c_{2P}$ are replaced by cubes $b_1 d, b_2 d, \cdots, b_P d$. The power saving for extracting $d$ is therefore given by:

$$\frac{C_0 V_{dd}^2}{2T_{cycle}} \quad \{ \quad (P-1)\sum_{k=1}^{K} n_{i_k} E_{i_k}(sw) +$$

$$\{\sum_{p=1}^{2P} E_{c_p}(sw) - \{PE_d(sw) + \sum_{i=1}^{2} E_{d_i}(sw) + \sum_{p=1}^{P} E_{b_p d}(sw)\} \quad \} \quad \}.$$

This can be easily extended to extracting double-cube divisors among multiple output Boolean functions as in [49].

As the active area (e.g., the number of literals) in a circuit strongly influences the power consumption, one must minimize a lexicographic cost $(\Delta_a, \Delta_p)$ where $\Delta_a$ is the literal saving factor and $\Delta_p$ is the power saving factor. At the same time, the above power saving factor is expensive to compute, therefore, it is desirable to calculate it only for a subset of candidate divisors (say, the top 10% of divisors in terms of their literal saving factors).

The above procedure ignores signal correlations due to reconvergent fanout. Techniques such as that in [28] or [18] can be used to calculate the activity factors more accurately. It will be also useful to extend the above approach to work under a real delay model. In particular, one can use the probability waveforms to calculate the power-saving cost of a candidate divisor.

## 3.4 Technology Decomposition

It is difficult to come up with a decomposed network which will lead to a minimum power implementation after technology mapping since gate loading and mapping information are unknown at this stage. Nevertheless, it has been 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 [64]. In the following, we describe algorithms for solving this problem for a fanout-free logic function (i.e. a function that has a tree realization.

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. This algorithm combines the two nodes with least weight iteratively until one node remains.

In a more general setting, a *weight combination function* $F(x, y)$ is used to produce the weight $W_i$ of internal node $i$ during tree construction. For each tree $T$, a *tree cost function*

$G(W_1, W_2, \ldots, W_{n-1})$ gives the cost. Parker [24] characterized a wide class of weight combination functions, for which Huffman's algorithm produces optimal trees under corresponding tree cost functions. If $F(x, y)$ does not satisfy these conditions, then Huffman's algorithm may not produce the optimal solution. The following $O(n^2 logn)$ greedy algorithm however produces very good results for problems under general weight combination functions: 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.

For *n-type* dynamic circuits, 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, the inverter output is 0 during the precharge period and the switching probability is given by:

$$W_o = w_{i_1} w_{i_2}$$

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

For *p-type* dynamic circuits, 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 is then given by:

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

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:

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

It can be shown that $W_o$ and $W_{\overline{o}}$ given above satisfy optimality conditions for Huffman's algorithm. Therefore, the tree decomposition for dynamic CMOS circuits (with uncorrelated input signals) can be solved optimally by using Huffman's algorithm.

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:

$$W_o = 2w_{i1}w_{i2}(1 - w_{i1}w_{i2}).$$

This function is not quasi-linear and therefore the tree decomposition problem for static CMOS circuits cannot be solved by using the Huffman's algorithm. Therefore, we resort to the greedy algorithm described earlier.

The weight combination functions for both dynamic and static CMOS circuits with correlated inputs are non-quasi-linear, and therefore, the same greedy algorithm must be used.

This work can be extended to the case where a probability waveform is given at each input of the node and the total switching activity must be optimized under a real delay model. The new problem does not lend itself to an optimal polynomial time algorithm (even for dynamic logic) as the tree cost is not monotone in the cost of its subtrees, that is, the minimum switching activity at the root of the tree is not necessarily obtained by minimizing the switching activity at the internal nodes. The greedy algorithm described above can however be extended to combine every pair of inputs and pick the pair which produces a probability waveform with minimum switching activity and repeat the above step until one node remains.

## 3.5 Technology Mapping

A successful and efficient solution to the minimum area mapping problem was suggested in [26] 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. [64] describes a technology mapping procedure for low power which is summarized next.

With each node in the network, we store a power-delay curve. A point on the power-delay curve represents the arrival time at the output of the node and the total power dissipation in its mapped transitive fanin cone up to (and including) the node. Points on the curve represent various mapping solutions with different tradeoffs between power dissipation and speed. We can drop point $P_1$ on the curve if there exists another point $P_2$ on the curve with lower power dissipation but equal or lower delay. This is possible because the solution associated

with $P_2$ is superior to the solution associated with $P_1$ in terms of power dissipation, delay or both. By dropping inferior points, the power-delay curve can always be made monotonically non-increasing without loss of optimality.

The technology mapping procedure consists of two graph traversal steps. Initially a postorder traversal of the NAND-decomposed network is performed, where for each node $n$ and for each gate $g$ matching at $n$ (a candidate match), a new power-delay curve is produced by appropriately merging the power-delay curves at the $inputs(n, g)$. The power-delay curves for successive gates $g$ matching at $n$ are then merged by applying a *lower-bound merge* operation on the corresponding power-delay curves. The power-delay curve computation and merging are performed recursively until a circuit output is reached. The set of $(t, p)$ pairs corresponding to the composite power-delay curve at the circuit output will define a set of arrival time-power dissipation tradeoffs for the user to choose from.

Given the required time $t$ at the circuit output, a suitable $(t, p)$ point on the power-delay curve for the circuit output is chosen. The gate $g$ matching at the circuit output which corresponds to this point and its inputs are thus identified. The required times $t_i$ at the inputs are computed from $t$, $g$, and the fact that these inputs must now drive gate $g$. The preorder traversal resumes at inputs of $g$ where $t_i$ is the constraining factor and a matching gate $g_i$ with minimum $p_i$ satisfying $t_i$ is sought.

The algorithm is optimal for trees and has polynomial run time on a node-balanced tree. It is easily extended to mapping a network modeled by a directed acyclic graph. Compared to a technology mapper that minimizes the circuit delay, this procedure leads to an average of 18% reduction in power consumption at the expense of 16% increase in area without any degradation in performance.

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 dynamic programming approach was adopted based on the assumption 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.

The extension to a real delay model is considered in [62]. Every point on the power-delay curve of a given node uniquely defines a mapped subnetwork from the circuit inputs up to the node. Again, the idea is to annotate each such point with the probability waveform for the node in the corresponding mapped subnetwork. Using this information, the total power cost

(due to steady-state transitions and hazards) of a candidate match can be calculated from the annotated power-delay curves at the inputs of the gate and the power-delay characteristics of the gate itself. The spatial correlations among the input waveforms can be captured using the tagging mechanism described previously.

## 3.6 Signal-to-Pin Assignment

It is necessary to construct a standard cell library where several variable-sized versions of a gate are available. These gates are sized with area minimization and input ordering in mind such that they give good delay response without high power dissipation.

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. It is well known that the signal to pin assignment in a CMOS logic gate has a sizable impact on the propagation delay through the gate [35, 10]. In particular, under the assumption that the input ramp time is small, the latest arriving signal should be assigned to the transistor near the output terminal of the gate. It is desirable to develop an assignment algorithm which minimizes dynamic power consumption on non-critical paths by signal reordering.

If we ignore the power dissipation due to charging and discharging of internal capacitances, it becomes obvious that high switching activity inputs should be matched with pins that have low input capacitance. However, the internal power dissipation also varies as a function of 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 therefore solve the following optimization problem:

$$\min_{\pi \in PP(n,g)} \sum_{i \in pins(g)} V_{dd}^2 C_i^G E_{\pi(i)}(sw) + \sum_{j \in internal\_nodes(g)} (V_{dd} - V_T)^2 C_j^{SD} E_{j,\pi}(charge/discharge)$$

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

As the number of functionally equivalent pins in a typical semi-custom library is not greater that six, it is feasible to exhaustively enumerate all pin permutations to find the

minimum power pin assignment. Alternatively, we can use heuristics, for example, a reasonable heuristic assigns the signal with largest probability of assuming a controlling value (zero for NMOS and one for PMOS) to the transistor near the output terminal of the gate. Alternatively, one could assign the signal with the earliest transition to a controlling value to the transistor near the output terminal. The rationale is that this transistor will switch off as often as (or as early as) possible, thus blocking the internal nodes from non-productive charge and discharge events.

Another heuristic [30] assigns the signal with highest switching activity to the input pin with the least capacitance. This is not very effective as in the semi-custom libraries, the difference in pin capacitances for logically equivalent pins is small.

The pin permutation for low power should take place on non-critical gates as it is in general different from the pin permutation for minimum delay.

# 4 Concluding Remarks

It has now become necessary to build new products with at least the same performance, but with demanding low power requirements. This has created a sudden change in design priorities leading to calls for low power design techniques, tools and methodologies. In this paper, we presented a set of techniques for power estimation and minimization at the logic level. Whenever possible, we also hinted directions for future research.

## Acknowledgement

This paper could have not been written without the help of students in the CAD group at USC (including, amongst others, C. Ding, S. Iman, R. Marculescu, D. Raiciu and C-Y. Tsui) whose research works are reflected here. The project was funded by the National Science Foundation and by a grant from the Intel Corporation.

# References

[1] K. Bartlett, R. K. Brayton, G. D. Hachtel, R. M. Jacoby, C. R. Morrison, R. L. Rudell, A. Sangiovanni-Vincentelli, and A. R. Wang. Multi-level logic minimization using implicit don't cares. In IEEE *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, volume 7, pages 723–740, June 1988.

[2] L. Benini, M. Favalli, and B. Ricco. Analysis of hazard contribution to power dissipation in CMOS IC's. In *Proceedings of the 1994 International Workshop on Low Power Design*, pages 27–32, April 1994.

[3] R. Brayton, R. Rudell, A. Sangiovanni-Vincentelli, and A. Wang. MIS: A multiple-level logic optimization system. IEEE *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, CAD-6(6), Nov. 1987.

[4] R. K. Brayton and C. McMullen. The decomposition and facorization of Boolean expressions. In *Proceedings of the International Symposium on Circuits and Systems*, pages 49–54, Rome, May 1982.

[5] R. Bryant. Graph-based algorithms for Boolean function manipulation. In IEEE *Transactions on Computers*, volume C-35, pages 677–691, August 1986.

[6] R. E. Bryant. MOSSIM: A switch-level simulator for MOS VLSI. In *Proceedings of the 18th Design Automation Conference*, pages 786–790, June 1989.

[7] 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.

[8] R. Burch, F. N. Najm, P. Yang, and T. Trick. A Monte Carlo approach for power estimation. IEEE *Transactions on VLSI Systems*, 1(1):63–71, March 1993.

[9] J. B. Burr. Stanford ultra low power CMOS. In *Proceedings of Hot Chips Symposium V*, pages 583–588, June 1993.

[10] B. S. Carlson and C-Y. R. Chen. Performance enhancement of CMOS VLSI circuits by transistor reordering. In *Proceedings of the 30th Design Automation Conference*, pages 361–366, June 1993.

[11] A. P. Chandrakasan, R. Allmon, A. Stratakos, and R. W. Brodersen. Design of portable systems. In *Proceedings of the* IEEE *Custom Integrated Circuits Conference*, May 1994.

[12] A. P. Chandrakasan, M. Potkonjak, J. Rabaey, and R. W. Brodersen. HYPER-LP: A system for power minimization using architectural transformation. In *Proceedings of the* IEEE *International Conference on Computer Aided Design*, pages 300–303, November 1992.

[13] A. P. Chandrakasan, S. S. Scheng, and R. W. Brodersen. Low power CMOS digital design. IEEE *Journal of Solid State Circuits*, 27(4):473–483, April 1992.

[14] C. Deng. Power analysis for CMOS/BiCMOS circuits. In *Proceedings of the 1994 International Workshop on Low Power Design*, pages 3–8, April 1994.

[15] S. Devadas, H-K. T. Ma, A. R. Newton, and A. Sangiovanni-Vincentelli. MUSTANG: State assignment of finite state machines targeting multi-level logic implementations. In IEEE *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, volume 7, pages 1290–1300, December 1988.

[16] C. Ding and M. Pedram. Efficient symbolic simulation under the extended bounded delay model. In *Proceedings of Tau'93: Int'l Workshop on Timing Issues in the Specification and Synthesis of Digital Systems*, September 1993.

[17] C. Ding and M. Pedram. Exact power analysis considering the impact of signal transition time. Technical report, Electrical Engineering-Systems Department, University of Southern California, March 1994.

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

[19] T. A. Fjeldly and M. Shur. Threshold voltage modeling and the subthreshold regime of operation of short-channel MOSFET's. IEEE *Transactions on Electron Devices*, 40(1):137–145, Jan. 1993.

[20] 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.

[21] L. H. Goldstein. Controllability/observability of digital circuits. IEEE *Transactions on Circuits and Systems*, 26(9):685–693, September 1979.

[22] S. Iman and M. Pedram. Multi-level network optimization for low power. In *Proceedings of the* IEEE *International Conference on Computer Aided Design*, pages 372–377, November 1994.

[23] Information Sciences Institute. MOSIS data sheet. Technical report, University of Southern California, 1992.

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

[25] S. M. Kang. Accurate simulation of power dissipation in VLSI circuits. IEEE *Journal of Solid State Circuits*, 21(5):889–891, Oct. 1986.

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

[27] S. Kirkpatrick, C. D. Gelatt, and M. P.Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, May 1983.

[28] B. Krishnamurthy and I. G. Tollis. Improved techniques for estimating signal probabilities. IEEE *Transactions on Computers*, 38(7):1245–1251, July 1989.

[29] H. M. Lieberstein. *A Course in Numerical Analysis*. Harper & Row Publishers, 1968.

[30] B. Lin and H. de Man. Low-power driven technology mapping under timing constraints. In *International Workshop on Logic Synthesis*, pages 9a.1–9a.16, April 1993.

[31] B. Lin and A. R. Newton. Synthesis of multiple-level logic from symbolic high-level description languages. In *IFIP International Conference on Very Large Scale Integration*, pages 187–196, August 1989.

[32] B. Lin and A. R. Newton. Synthesis of multiple-level logic from symbolic high-level description languages. In *IFIP International Conference on Very Large Scale Integration*, August 1989.

[33] D. Liu and C. Svensson. Trading speed for low power by choice of supply and threshold voltages. IEEE *Journal of Solid State Circuits*, 28(1):10–17, January 1993.

[34] R. Marculescu, D. Marculescu, and M. Pedram. Logic level power estimation considering spatiotemporal correlations. In *Proceedings of the* IEEE *International Conference on Computer Aided Design*, pages 294–299, November 1994.

[35] M. Marek-Sadowska and S. P. Lin. Pin assignment for improved performance in standard cell design. In *Proceedings of the International Conference on Computer Design*, pages 339–342, 1990.

[36] G. De Micheli, R. K. Brayton, and A. Sangiovanni-Vincentelli. Optimal state assignment of finite state machines. In IEEE *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, volume 4, pages 269–285, July 1985.

[37] J. Monteiro, S. Devadas, and A. Ghosh. Estimation of switching activity in sequential logic circuits with applications to synthesis for low power. In *Proceedings of the 31st Design Automation Conference*, pages 12–17, June 1994.

[38] J. Monteiro, S. Devadas, B. Lin, C-Y. Tsui, M. Pedram, and A. M. Despain. Exact and approximate methods of switching activity estimation in sequential logic circuits. In *Proceedings of the 1994 International Workshop on Low Power Design*, pages 117–122, April 1994.

[39] R. S. Muller and T. I. Kamins. *Device electronics for integrated circuits.* John Wiley & Sons, 1986.

[40] S. Muroga, Y. Kambayashi, H.C. Lai, and J.N. Culliney. The transduction method - design of logic networks based on permissible functions. In IEEE *Transactions on Computers*, 1989.

[41] L. W. Nagel. SPICE2: A computer program to simulate semiconductor circuits. Technical Report UCB/ERL M75/520, Electronics Research Lab, University of California at Berkeley, 1975.

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

[43] 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*, 9(4):439–450, April 1990.

[44] Y. Nakagome, K. Itoh, M. Isoda, K. Takeuchi, and M. Aoki. A sub-1-V swing bus architecture for future low power ULSIs. IEEE *Journal of Solid State Circuits*, 28(4):414–419, April 1993.

[45] E. Olson and S. M. Kang. Low-power state assignment for finite state machines search. In *Proceedings of the 1994 International Workshop on Low Power Design*, pages 63–68, April 1994.

[46] P. Van Oostende, P. Six, J. Vandewalle, and H. De Man. Estimation of typical power of synchronous CMOS circuits using a hierarchy of simulators. IEEE *Journal of Solid State Circuits*, 28(1):26–39, Jan. 1993.

[47] John K. Ousterhout. Crystal: A Timing Analyzer for nMOS VLSI Circuits. In *Third Caltech VLSI Conference*, 1983.

[48] K. P. Parker and J. McCluskey. Probabilistic treatment of general combinational networks. IEEE *Transactions on Computers*, C-24:668–670, Jun. 1975.

[49] J. Rajski and J. Vasudevamurthy. The testability-preserving concurrent decomposition and factorization of Boolean expressions. IEEE *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 11(6):778–793, June 1993.

[50] K. Roy and S. C. Prasad. Circuit activity based logic synthesis for low power reliable operations. IEEE *Transactions on VLSI Systems*, 1(4):503–513, December 1993.

[51] A. Salz and M. A. Horowitz. IRSIM: An incremental MOS switch-level simulator. In *Proceedings of the 26th Design Automation Conference*, pages 173–178, June 1989.

[52] J. Savir, G. Ditlow, and P. Bardell. Random pattern testability. IEEE *Transactions on Computers*, 33(1):1041–1045, jan 1984.

[53] H. Savoj, R. K. Brayton, and H. J. Touati. Extracting local don't cares for network optimization. In *Proceedings of the* IEEE *International Conference on Computer Aided Design*, pages 514–517, November 1991.

[54] P. Schneider and U. Schlichtmann. Decomposition of boolean functions for low power based on a new power estimation technique. In *Proceedings of the 1994 International Workshop on Low Power Design*, pages 123–128, April 1994.

[55] S. C. Seth and V. D. Agrawal. *A new model for claculation of probabilistic testability in combinational circuits.* Elsevier Science Publishers, 1989.

[56] S.C. Seth, L. Pan, and V.D. Agrawal. PREDICT - Probabilistic estimation of digital circuit testability. In *Proceedings of the Fault Tolerant Computing Symposium*, pages 220–225, June 1985.

[57] 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.

[58] K. Shimohigashi and K. Seki. Low-voltage ULSI design. IEEE *Journal of Solid State Circuits*, 28:408–413, April 1993.

[59] G. I. Stamoulis and I. N. Hajj. Improved techniques for probabilistic simulation including signal correlation effects. In *Proceedings of the 30th Design Automation Conference*, pages 379–383, June 1993.

[60] C-L. Su, C-Y. Tsui, and A. M. Despain. Low power architecture design and compilation techniques for high-performance processors. In *CompCon'94 Digest of Technical Papers*, pages 489–498, February 1994.

[61] C-Y. Tsui, M. Pedram, C-H. Chen, and A. M. Despain. Low power state assignment targeting two- and multi-level logic implementations. In *Proceedings of the* IEEE *International Conference on Computer Aided Design*, pages 82–87, November 1994.

[62] C-Y. Tsui, M. Pedram, and A. M. Despain. Efficient estimation of dynamic power dissipation under a real delay model. In *Proceedings of the* IEEE *International Conference on Computer Aided Design*, pages 224–228, November 1993.

[63] C-Y. Tsui, M. Pedram, and A. M. Despain. Power estimation considering charging and discharging of internal nodes of cmos gates. In *Proc. the Synthesis and Simulation Meeting and Int'l Interchange*, pages 345–354, October 1993.

[64] C-Y. Tsui, M. Pedram, and A. M. Despain. Technology decomposition and mapping targeting low power dissipation. In *Proceedings of the 30th Design Automation Conference*, pages 68–73, June 1993.

[65] C-Y. Tsui, M. Pedram, and A. M. Despain. Exact and approximate methods for calculating signal and transition probabilities in fsms. In *Proceedings of the 31st Design Automation Conference*, pages 18–23, June 1994.

[66] A. Tyagi. Hercules: A power analyzer of MOS VLSI circuits. In *Proceedings of the* IEEE *International Conference on Computer Aided Design*, pages 530–533, November 1987.

[67] 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*, 19:468–473, August 1984.

[68] T. Villa and A. Sangiovanni-Vincentelli. NOVA: State assignment of finite state machines for optimal two-level logic implementations. In IEEE *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, volume 9, pages 905–924, September 1990.

[69] S. Yang and M. Ciesielski. On the relationship between input encoding and logic minimization. In *Proceedings of the Twenty Third Hawaii International Conference on the System Sciences*, volume I, pages 377–386, January 1990.