# 3

# Power Analysis and Optimization from Circuit to Register-Transfer Levels

Jose Monteiro
*INESC-Instituto de Engenharia de Sistems e Computadores*
*Lisboa, Portugal*

Rakesh Patel
*Intel Corp.*
*Chandler, Arizona*

Vivek Tiwari
*Intel Corp.*
*Santa Clara, California*



## 3.1 Introduction

The increasing speed and complexity of today's designs implies a significant increase in the power consumption of very large-scale integration (VLSI) circuits in the near future. To meet this challenge, we need to develop design techniques to reduce power. The complexity of today's ICs with over 100 million transistors, clocked at over 1 GHz, demands computer-aided design (CAD) tools and methodologies for a successful design in time to market.

One of the key features that led to the success of complementary metal-oxide semiconductor (CMOS) technology was its intrinsic low-power consumption. This meant that circuit designers and electronic design automation (EDA) tools could afford to concentrate on maximizing circuit performance and minimizing circuit area. Another interesting feature of CMOS technology is its nice scaling properties, which has permitted a steady decrease in the feature size, allowing for more and more complex systems on a single chip, working at higher clock frequencies.

Power consumption concerns came into play with the appearance of the first portable electronic systems in the late 1980s. In this market, battery lifetime is a decisive factor for the commercial success of

the product. Another fact that became apparent at about the same time was that the increasing integration of more active elements per die area would lead to prohibitively large-energy consumption of an integrated circuit. A high absolute level of power is not only undesirable for economic and environmental reasons, but it also creates the problem of heat dissipation. In order to keep the device working at acceptable temperature levels, excessive heat may require expensive heat removal systems.

These factors have contributed to the rise of power as a major design parameter on par with performance and die size. In fact, power consumption is regarded as the limiting factor in the continuing scaling of CMOS technology.

To respond to this challenge, in the last decade or so, intensive research has been put into developing computed-aided design (CAD) tools that address the problem of power optimization. Initial efforts were directed to circuit and logic-level tools because at this level CAD tools were more mature and there was a better handle on the issues. Today, most of the research for CAD tools targets system or architectural level optimization, which potentially have a higher overall impact, given the breadth of their application. Together with optimization tools, efficient techniques for power estimation are required, both as an absolute indicator that the circuit's consumption meets some target value and as a relative indicator of the power merits of different alternatives during design space exploration.

This chapter provides an overview of the most significant CAD techniques proposed for low power. We start in Section 3.2 by describing the issues and methods for power estimation at different levels of abstraction, thus defining the targets for the tools presented in the following sections. In Sections 3.3 and 3.4, we review power optimization techniques at the circuit and logic levels of abstraction respectively.

## 3.2 Power Analysis

Given the importance of the power consumption parameter in circuit design, EDA tools are required to provide power estimates for a design. When evaluating different design alternatives, we need estimates that indicate the most effective alternative in terms of power. Since estimates may be required for multiple alternatives, often iteratively, and relative accuracy suffices, absolute power accuracy is sometimes sacrificed for tool response speed. Second, an accurate absolute power consumption estimate is required before fabrication to guarantee that the circuit meets the allocated power budget.

Obtaining a power estimate is significantly more complex than circuit area estimates, or even delay estimates, because power is related not only to the circuit topology, but also to the activity of the signals.

Typically, design exploration is performed at each level of abstraction, thus creating the need for power estimation tools at the different levels. The higher the abstraction level, the less the information there is about the actual circuit implementation, implying less assurance about the power estimate accuracy.

In this section, we first discuss the components of power consumption in CMOS circuits. We then discuss how each of these components is estimated at the different design abstraction levels.

### 3.2.1 Power Components in CMOS Circuits

Power consumption of digital CMOS circuits is generally considered in terms of three components [1]:

- Dynamic power ($P_{\text{dyn}}$)
- Short-circuit power ($P_{\text{short}}$)
- Static power ($P_{\text{static}}$)

The total power consumption is given by the sum of these components:

$$P = P_{\text{dyn}} + P_{\text{short}} + P_{\text{static}} \tag{3.1}$$

The dynamic power component, $P_{\text{dyn}}$, is related to the charging and discharging of the load capacitance at the gate output, $C_{\text{out}}$. This is a parasitic capacitance that can be lumped at the output of the gate. Today, this component is still the dominant source of power consumption in a CMOS gate.

As an illustrative example, consider the inverter circuit depicted in Figure 3.1 (to form a generic CMOS gate, the bottom transistor, nMOS, can be replaced by a network of nMOS transistors, and the top transistor, pMOS, by a complementary network of pMOS transistors). When the input goes low, the nMOS transistor is cutoff and the pMOS transistor conducts. This creates a direct path between the voltage supply and $C_{out}$. Current $I_p$ flows from the supply to charge $C_{out}$ up to the voltage level $V_{dd}$. The amount of charge drawn from the supply is $C_{out}V_{dd}$ and the energy drawn from the supply equals $C_{out}V_{dd}^2$. The energy actually stored in the capacitor, $E_c$, is only half of this, i.e., $E_c = 1/2C_{out}V_{dd}^2$. The other half is dissipated in the resistance represented by the pMOS transistor. During the subsequent low to high input transition, the pMOS transistor is cutoff and the nMOS transistor conducts. This connects the capacitor $C_{out}$ to the ground, leading to the flow of current $I_n$. $C_{out}$ discharges and its stored energy, $E_c$, is dissipated in the resistance represented by the nMOS transistor. Therefore, an amount of energy equal to $E_c$ is dissipated (consumed) every time the output makes a transition. Given the number of transitions $N$ that a gate makes in a period of time $T$, its dynamic power consumption during that time period is given by

$$P_{dyn} = E_c N/T = \tfrac{1}{2} C_{out} V_{dd}^2 N/T \tag{3.2}$$

In the case of synchronous circuits, an estimate, $\alpha$, of the average number of transitions the gate makes per clock cycle, $T_{clk} = 1/f_{clk}$, can be used to compute average dynamic power

$$P_{dyn} = E_c \alpha f_{clk} = \tfrac{1}{2} C_{out} V_{dd}^2 \alpha f_{clk} \tag{3.3}$$

$C_{out}$ is the sum of three components, $C_{int}$, $C_{wire}$, and $C_{load}$. Of these, $C_{int}$ represents the internal capacitance of the gate. This basically consists of the diffusion capacitance of the drain regions connected to the output. $C_{load}$ represents the sum of gate capacitances of the transistors this logic gate is driving; and $C_{wire}$ the parasitic capacitance of the wiring used to interconnect the gates, including the capacitance between the wire and the substrate, the capacitance between neighboring wires, and the capacitance due to the fringe effect of electric fields. The term $\alpha C_{out}$ is generally called the switched capacitance, which measures the amount of capacitance that is charged or discharged in one clock cycle.

The short-circuit power component, $P_{short}$, is also related to the switching of the output of the gate. During the transition of the input line from one voltage level to the other, there is a period of time when both the pMOS and the nMOS transistors are on, thus creating a path from $V_{dd}$ to ground. Thus, each time a gate switches, some amount of energy is consumed by the current, indicated as $I_{short}$ in Figure 3.1, that flows through both transistors during this period. The short-circuit power is determined by the time the input voltage $V_{in}$ remains between $V_{Tn}$ and $V_{dd} - V_{Tp}$, where $V_{Tn}$ and $V_{Tp}$ are the threshold voltages of the nMOS and the pMOS transistors, respectively. Careful design to minimize low slope input ramps, namely through the appropriate sizing of the transistors, can keep this component as a small fraction of the total power, and hence it is generally considered only as a second-order effect. Given an estimate of the average amount of charge, $Q_{short}$, which is carried by the short-circuit current per output transition, the short-circuit power is obtained with

$$P_{short} = Q_{short} V_{dd} \alpha f_{clk} \tag{3.4}$$

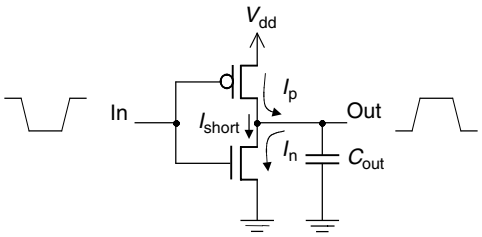


**FIGURE 3.1** Illustration of the dynamic and short-circuit power components.

The static power component, $P_{static}$, is due to leakage currents in the MOS transistors. As the name indicates, this component is not related to the circuit activity and exists as long as the circuit is powered. The source and drain regions of a MOS transistor (MOSFET) can form reverse-biased parasitic diodes with the substrate. There is leakage current associated with these diodes. This current is very small and used to be negligible compared to dynamic power consumption. Another type of leakage current occurs due to the diffusion of carriers between the source and drain even when the MOSFET is in the cutoff region, i.e., when the magnitude of the gate-source voltage, $V_{GS}$, is below the threshold voltage, $V_T$. In this region, the MOSFET behaves like a bipolar transistor and the subthreshold current is exponentially dependent on $V_{GS} - V_T$.

Another situation that can lead to static power dissipation in CMOS is when a degraded voltage level (e.g., the "high" output level of an nMOS pass transistor) is applied to the inputs of a CMOS gate. A degraded voltage level may leave both the nMOS and pMOS transistors in a conducting state, leading to continuous flow of short-circuit current. This again is undesirable and care should be taken to avoid it in practice.

The above is true for pure CMOS design styles. In certain specialized circuits, namely for performance reasons, alternative design styles may be used. Some design styles produce a current when the output is constant at one voltage level, thus contributing to the increase in static power consumption. One example is the domino design style, where a precharged node needs to be recharged on every clock cycle if the output of the gate happens to be the opposite of the precharge value. Another example is the pseudo-nMOS logic family, where the pMOS network of a CMOS gate is replaced by a single pMOS transistor that always conducts. This logic style will have a constant current flowing at all times that the output is at logic 0, i.e., when there is a direct path to ground through the nMOS network.

### 3.2.2 Analysis at the Circuit Level

Power estimates at the circuit level are generally obtained using a circuit-level simulator such as Spice [2]. Given a user-specified representative sequence of input values, the simulator solves the circuit equations to compute voltage and current waveforms at all nodes in the electrical circuit. By averaging the current values drawn from the source, $I_{avg}$, the simulator can output the average power consumed by the circuit, $P = I_{avg} V_{dd}$ (if multiple power sources are used, the total average power will be the sum of the power drawn from the different power sources).

At this level, complex models for the circuit devices can be used. These models allow for the accurate computation of the three components of power — dynamic, short, and static power. Since the circuit is described at the transistor level, correct estimates can be computed not only for CMOS, but also for any logic design style and even analog modules. If the layout and routing of the circuit has been performed, and the simulation is made on the back-annotated circuit description, i.e., with realistic interconnect capacitive and resistive values, the power estimates obtained can be very close to those of the actual fabricated circuit.

The problem is that the simulation process with this level of detail implies the solution of complex systems of equations, hence creating severe limitations in terms of the size of the circuit that can be handled. Another limitation is that the length of the input sequences must necessarily be very short since simulation is very time consuming. The consequence of this is that power estimates may not reflect the real statistics of the inputs well.

For these reasons, full-fledged circuit-level power estimation is typically performed only for the accurate characterization of small circuit modules.

An approach to permit the applicability of the circuit-level simulation to larger designs is to resort to very simple models for the active devices. Naturally, this simplification comes at the cost of some accuracy loss.

Switch-level simulation is a limiting case, where transistor models are simply reduced to switches, which can be either opened or closed, with some associated parasitic resistive and capacitive values. This approach makes simulation run much more efficiently, allowing for the estimation of significantly larger circuit modules under much longer input sequences. Switch-level simulation can still model with fair accuracy the dynamic and short circuit components of power, but this is no longer true for leakage power. Although a few years ago designers were willing to ignore this power component since it would make for a negligible fraction of total power, its relative importance is increasing. Leakage-power estimations must

then be performed independently using specifically tailored tools. Many different approaches for leakage-power estimation have been proposed recently, which are presented in the next section.

Intermediate complexity solutions exist. For example, PowerMill [3] is a circuit-level power estimation tool from Synopsys, Inc, which employs table lookup of current models for given transistor sizes. Furthermore, its algorithm uses circuit partitioning and solves the circuit equations independently on each partition. Although some error is introduced because interactions between different partitions are not accounted for, this technique greatly simplifies the problem to be solved, allowing for fast circuit-level simulation of large designs.

### 3.2.3 Static Power Estimation

Static power analysis is typically performed using the sub-threshold model to estimate leakage per micron of gate width of a minimum-length transistor. Then that model is extended to estimate leakage over the entire chip. Typically, the stacking factor (leakage reduction from stacking of devices) is a first-order component of this extension and serves to modify the total effective width of devices under analysis [4]. Analysis can be viewed as the modification of this total width by the stacking factor.

Most analytical works on leakage have used the BSIM2 sub-threshold current model [5],

$$I_{\text{sub}} = A\exp\left(\frac{(V_{\text{GS}} - V_{\text{T}} - \gamma' V_{\text{SB}} + \eta V_{\text{DS}})}{nV_{\text{TH}}}\right)\left(1 - e^{\frac{V_{\text{DS}}}{V_{\text{TH}}}}\right) \tag{3.5}$$

where $V_{\text{GS}}$, $V_{\text{DS}}$, and $V_{\text{SB}}$ are the gate-source, drain-source, and source-bulk voltages, respectively, $V_{\text{T}}$ the zero bias threshold voltage, $V_{\text{TH}}$ the thermal voltage ($kT/q$), $\gamma'$ the linearized body-effect coefficient, $\eta$ is the drain induced barrier lowering (DIBL) coefficient and

$$A = \mu_0 C_{\text{ox}} (W/L_{\text{eff}}) V_{\text{TH}}^2 e^{1.8}$$

The BSIM2 leakage model incorporates all the leakage behavior that we are presently concerned with. In summary, it accounts for the exponential increase in leakage with reduction in threshold voltage and gate-source voltage. It also accounts for the temperature dependence of leakage.

Calculating leakage current by applying Equation (3.5) to every single transistor in the chip can be very time-consuming. To overcome this barrier, empirical models for dealing with leakage at a higher level of abstraction have been studied [6,7]. For example, a simple empirical model is as follows [7]:

$$I_{\text{leak}} = I_{\text{off}} \frac{W_{\text{tot}}}{X_{\text{s}}} X_{\text{t}} \tag{3.6}$$

where $I_{\text{off}}$ is the leakage current per micron of a single transistor measured from actual silicon at a given temperature, $W_{\text{tot}}$ the total transistor width (sum of all N and P devices), $X_{\text{s}}$ an empirical stacking factor based on the observation that transistor stacks leak less than single devices. $X_{\text{t}}$ is the temperature factor and is used to scale $I_{\text{off}}$ to the appropriate junction temperature of interest. The $I_{\text{off}}$ value is typically specified at room temperature (therefore the need for a temperature factor to translate to the temperature of interest).

### 3.2.4 Logic-Level Power Estimation

A key observation made in Section 3.2.1 that makes power estimation at the logic level feasible is that if the input of the gate rises fast enough, the energy consumed by each output transition does not depend on the resistive characteristics of the transistors and is simply a function of the capacitive load, $C_{\text{out}}$, the gate is driving, i.e., $E_{\text{c}} = 1/2 C_{\text{out}} V_{\text{dd}}^2$. Given parasitic gate and wire capacitance models that allow the computation of $C_{\text{out\_}i}$ for each gate $i$ in a gate-level description of the circuit, power estimation at the logic level reduces to computing the number of transitions that each gate makes in a given period of time, i.e., the switching activity of the gate. This corresponds to either parameter $N$ or $\alpha$, and we need only apply Equation (3.2) or Equation (3.3), respectively, to obtain power.

Naturally, this estimate refers only to the dynamic power component. Although this component used to make for over 90% of total power, current estimates for total power consumption must take leakage power into account, meaning that the methods described in the previous section must complement the logic-level estimate.

In many cases, power estimates at the logic level serve as indicators for guiding logic-level power optimization techniques, which typically target the dynamic power reduction, hence only an estimate for this component is required.

There are two classes of techniques for the switching activity computation — simulation-based and probabilistic analysis (also known as dynamic and static techniques, respectively).

### 3.2.4.1 Simulation-Based Techniques

In simulation-based switching activity estimation, highly optimized logic simulators are used, allowing for a fast simulation of a large number of input vectors. This approach raises two main issues: the number of input vectors to simulate and the delay model to use for the logic gates.

The simplest approach to model the gate delay is to assume zero delay for all the gates, meaning that all transitions in the circuit occur at the same time instant. As a consequence, each gate has at most one transition per input vector. In reality, logic gates have a nonzero transport delay, which may lead to different arrival times of transitions at the inputs of a logic gate due to different signal propagation paths. As a consequence, the output of the gate may switch multiple times in response to a single input vector. An illustrative example is shown in Figure 3.2.

Consider that initially signal $x$ is set to 1 and signal $y$ is set to 0, meaning that both signals $w$ and $z$ are set to 1. If $y$ makes a transition to 1, then $z$ will first respond to this transition by switching to 0. However, at about the same time, $w$ is switching to 0, hence causing $z$ to switch back to 1.

This spurious activity can make for a significant fraction of the overall switching activity, which in the case of circuits with a high degree of re-convergent signals, such as multipliers, may be above 50%. The modeling of gate delays in logic-level power estimation is thus of crucial significance. For an accurate switching activity estimate, the simulation must use a general delay model where gate delays are retrieved from a precharacterized library of gates.

The second issue is determining the number of input vectors to simulate. If the objective is to obtain a power estimate of a logic circuit under a user-specified, potentially long sequence of input vectors, then the switching activity can be easily obtained through logic simulation. When what is available are some statistics for the inputs, then this sequence of input vectors needs to be generated. One option is to generate a sequence of input vectors that approximates the given input statistics and simulate until the average power converges, that is, until this value stays within a margin $\varepsilon$ during the last $n$ input vectors, where $\varepsilon$ and $n$ are user-defined parameters.

An alternative is to compute beforehand the number of input vectors required for a given allowed percentage error $\varepsilon$ and confidence level $\theta$. Under a basic assumption that the power consumed by a circuit over a period of time $T$ has a Normal distribution, the approach described in [8] uses the Central Limit Theorem to determine the number of input vectors with which to simulate the circuit

$$N \geq \left( \frac{z_{\theta/2} s}{\overline{p} \varepsilon} \right)^2$$

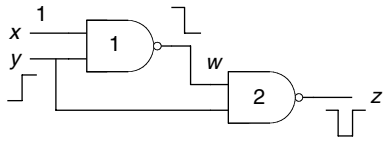


**FIGURE 3.2** Example of a logic circuit with glitching and spatial correlation.

where $N$ is the number of input vectors, $\overline{p}$ and $s$ the measured average and standard deviation of the power, and $z_{\theta/2}$ is obtained from the Normal distribution.

In practice, for typical combinational circuits and reasonable error and confidence levels, the number of input vectors that need to be simulated to obtain the overall average switching activity is typically very small (of the order of thousands) even for complex logic circuits. However, in many situations, accurate average switching activity for each of the nodes in the circuit is required. A high level of accuracy for low-switching nodes may require a prohibitively large number of input vectors. The designer may need to relax the accuracy in these nodes, considering that these nodes have less impact in the dynamic power consumption of the circuit.

#### 3.2.4.2 Probabilistic Techniques

The idea behind probabilistic techniques is to propagate directly the input statistics to obtain the switching probability of each node in the circuit. This approach is potentially very efficient, as only a single pass through the circuit is needed. However, it requires a new simulation engine with a set of rules for propagating the signal statistics. For example, the probability that the output of an AND gate evaluates to 1 is the intersection of the conditions that set each of its inputs to 1. If the inputs are independent, then this is just the multiplication of the probability that each input evaluates to 1. Similar rules can be derived for any logic gate and for different statistics, namely transition probabilities. Although all of these rules are simple, there is a new set of complex issues to be solved.

One of them is the delay model, as discussed above. Under a general delay model, each gate may switch at different time instants in response to a single-input change. Thus, we need to compute switching probabilities for each of these time instants. Assuming transport delays $\Delta_1$ and $\Delta_2$ for the gates in the circuit of Figure 3.2, this means that signal $z$ will have some probability of making a transition at instant $\Delta_2$ and some other probability of making a transition at instant $\Delta_1 + \Delta_2$. Naturally, the total switching activity of signal $z$ will be the sum of these two probabilities.

Another issue is spatial correlation. When two logic signals considered together, they can only be assumed to be independent if they do not have any common input signal in their support. If there is one or more common input, we say that these signals are spatially correlated. To illustrate this point, consider again the logic circuit of Figure 3.2 and assume that both input signals $x$ and $y$ are independent and have a $p_x = p_y = 0.5$ probability of being at 1. Then $p_w$, the probability that $w$ is 1, is simply $p_w = 1 - p_x p_y = 0.75$. However, it is not true that $p_z = 1 - p_w p_y = 0.625$, because signals $w$ and $y$ are not independent: $p_w p_y = (1 - p_x p_y) p_y = p_y - p_x p_y$ (note that $p_y p_y = p_y$), with $p_z = (1 - p_y + p_x p_y) = 0.75$. This indicates that not accounting for spatial correlation can lead to significant errors in the calculations.

Input signals may also be spatially correlated. Yet, in many practical cases, this correlation is ignored, either because it is simply not known or because of the difficulty in modeling this correlation. For a method that is able to account for all types of correlations among signals, see [9], but its complexity is such that it cannot be applied to very large designs.

A third important issue is temporal correlation. In probabilistic methods, the average switching activity is computed from the probability of a signal making a transition 0 to 1 or 1 to 0. Temporal correlation measures the probability that a signal is 0 or 1 in the next instant, given that its present value is 0 or 1. This means that computing the static probability of a signal being 1 is not sufficient; we need to calculate the transition probabilities directly so that temporal correlation is taken into account. Consider signals $x$ and $y$ in Figure 3.3, where the vertical lines indicate clock periods. The number of periods where these two signals are 0 or 1 is the same, hence the probability of the signals being at 1 is $p_x^1 = p_y^1 = 0.5$ (and the probability being at 0 is $p_x^0 = p_y^0 = 0.5$). If we only consider this parameter, thus ignoring temporal correlation, the transition probability for both signals is the same and can be computed as $\alpha = p^{01} + p^{10} = p^0 p^1 + p^1 p^0 = 0.5$. However, we can see that during the depicted time interval, signal $x$ stays low for three clock cycles, stays high for another three cycles, and has a single clock cycle with a rising transition and another with a falling transition. Averaging over the number of clock periods, we have $p_x^{00} = 3/8 = 0.375$, $p_x^{01} = 1/8 = 0.125$, $p_x^{10} = 1/8 = 0.125$, and $p_x^{11} = 3/8 = 0.375$. Therefore,

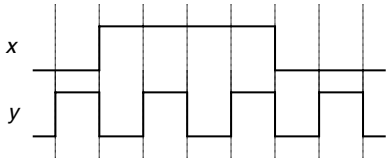


**FIGURE 3.3** Example signal to illustrate the concept of temporal correlation.

the actual average switching activity of $x$ is $\alpha_x = p_x^{01} + p_x^{10} = 0.25$. As for signal $y$, it never stays low or high, making a transition on every clock cycle. Hence, $p_y^{00} = p_y^{11} = 0$ and $p_y^{01} = p_y^{10} = 4/8 = 0.5$, and the actual average switching activity of $y$ is $\alpha_y = p_y^{01} + p_y^{10} = 1.0$. This example illustrates the importance of modeling temporal correlation and indicates that probabilistic techniques need to work with transition probabilities for accurate switching activity estimates.

It has been shown that an exact modeling of these issues makes the computation of the average switching activity an NP-complete problem, meaning that exact methods are only applicable to small circuits, thus of little practical interest. Many different approximation schemes have been proposed [10].

#### 3.2.4.3 Sequential Circuits

Computing the switching activity for sequential circuits is significantly more difficult because we need to ensure that the state space has been visited in a representative manner. For simulation-based methods, this requirement may imply too large an input sequence and, in practice, convergence is hard to guarantee.

Probabilistic methods can be effectively applied to sequential circuits as the statistics for the state lines can be derived from the circuit. The exact solution would require the computation of the transition probabilities between all pairs of states in the sequential circuit. In many cases, enumerating the states in the circuit is not possible, since these are exponential in the number of latches in the circuit. A common approximation is to compute the transition probabilities for each state line [11]. To recover to some degree the spatial correlation between state lines, a typical approach is to duplicate the subset of logic that generates the next state signals and append it to the present state lines, as illustrated in Figure 3.4. Then the method for combinatorial circuits is applied on this modified network, ignoring the switching activity in the duplicated next state logic block.

### 3.2.5 Analysis at the Register-Transfer Level

At the register-transfer level (RTL), the circuit is described in terms of a set of interconnected modules of varied complexity, from simple logic gates to full-blown multipliers. Power estimation at this level basically consists of determining the signal statistics at the input of each of these modules and then feeding these values to the module's power model to evaluate its dissipation. These models are normally available from the library of modules. One way to obtain these power models is to characterize the module using logic or circuit-level estimators; this process is known as macro-modeling. We refer to Chapters 7 and 13, vol.1 of this handbook, where this topic is discussed in more detail.

## 3.3 Circuit-Level Power Optimization

From the equations that model power consumption, it is easy to observe that reduction of the supply voltage has the largest impact on power reduction, given its quadratic dependency. This has been the largest source of power reductions and is widely applied across the VLSI industry. However, unless accompanied by the appropriate process scaling, reducing $V_{dd}$ comes at the cost of increased propagation delays, necessitating the use of techniques to recover the lost performance.

Lowering the frequency of operation, $f_{clk}$, also reduces power consumption. This may be an attractive option in situations with low-performance requirements. Yet the power efficiency of the circuit is not improved, as the amount of energy per operation remains constant.

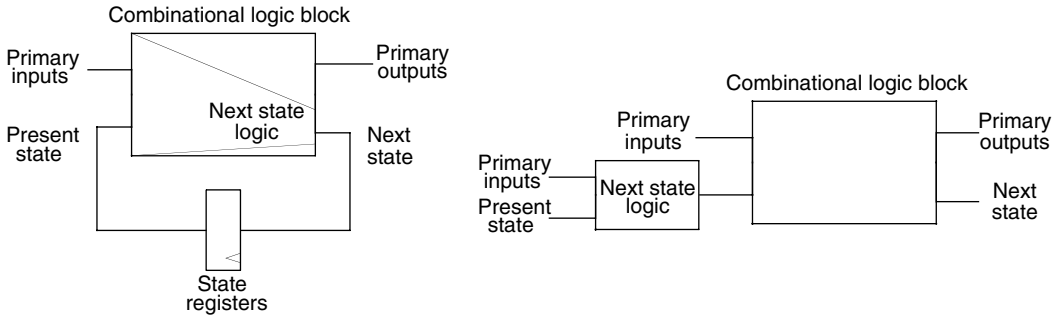


**FIGURE 3.4** Creating temporal and spatial correlation among state lines.

A more interesting option is to address the switched capacitance term, $\alpha C_{out}$, by redesigning the circuit such that the overall switching activity is reduced, or the overall circuit capacitance is reduced, or by a combination of both, where the switching activity in high-capacitance nodes is reduced possibly by exchanging this with higher switching in nodes with lower capacitance.

In the following, we discuss techniques at different levels of abstraction that have been developed to address each of these points.

### 3.3.1 Transistor Sizing

A large part of high-performance CPUs is typically custom-designed. These designs typically involve manual tweaking of transistors to upsize drivers in critical paths. If too many transistors are upsized unnecessarily, certain designs can operate on the steep part of a circuit's power-delay curve. In addition, the choice of logic family used (e.g., static vs. dynamic logic) can also greatly influence the circuit's power consumption. The traditional emphasis on performance often leads to over-design that is wasteful for power. An emphasis on lower power, however, motivates identification of such sources of power wastage. An example of this is the case where paths that are designed faster than they ultimately need to be. For synthesized blocks, the synthesis tool can automatically reduce power by downsizing devices in such paths. For manually designed blocks, on the other hand, downsizing may not always be done. Automated downsizing tools can thus have a big impact. The benefit of such tools is power savings as well as productivity enhancement over manual designs. Many custom designers are now exploring multiple-$V_T$ techniques to take greater advantage of the transistor sizing. The main idea here is to use lower $V_T$ transistors in critical paths rather than large high-$V_T$ transistors. The main issue with this technique is the increase in subthreshold leakage due to low $V_T$. So it is very important to use low-$V_T$ transistors selectively and to optimize their usage to achieve a good balance between capacitive current and leakage current in order to minimize total current.

### 3.3.2 Voltage Scaling, Voltage Islands, Variable $V_{dd}$

Power dissipation consists of a static component and a dynamic component. Since dynamic power is proportional to the square of the supply voltage ($V_{dd}$), the reduction in $V_{dd}$ is the most effective way for reducing power. The industry has thus steadily moved to lower $V_{dd}$. Indeed, reducing the supply voltage is the best for low-power operation, even after taking into account the modifications to the system architecture, which are required to maintain the computational throughput. Another issue with voltage scaling is that to maintain performance, threshold voltage ($V_T$) also needs to be scaled down since circuit speed is roughly inversely proportional to ($V_{dd} - V_T$). Typically, $V_{dd}$ should be higher than $4V_T$ if speed is not to suffer excessively. As the threshold voltage decreases, sub-threshold leakage current increases exponentially. At present, $V_T$ is high enough that sub-threshold current is dominated by the dynamic component of the total active current, which dominates the total standby current (including gate and well

leakage components). However, with every 0.1 V reduction in $V_T$, sub-threshold current increases ten times. In nanometer technologies, with further $V_T$ reduction, sub-threshold current will become a significant portion, or even a dominant portion, of the overall chip current. At sub-0.1 μm feature sizes, the leakage power starts eating into the benefits of lower $V_{dd}$. In addition, design of dynamic circuits, caches, sense-amps, PLAs, etc., becomes difficult at higher sub-threshold leakage currents. Lower $V_{dd}$ also exacerbates noise and reliability concerns. To combat sub-threshold current increase, various techniques have been developed.

Voltage islands and variable $V_{dd}$ are variations of voltage scaling that can be used at the lower level. Voltage scaling is mainly technology-dependent and typically applied to the whole chip. Voltage islands are more suitable for system-on-chip (SOC) designs, which integrate different functional modules with various performance requirements on a single chip. We refer the reader to the RTL power analysis and optimization techniques chapter (chapter 13 of vol.1 of this handbook) for more details on the voltage island technique. The variable voltage and voltage-island techniques are complementary and can be implemented on the same block and used simultaneously. In the variable voltage technique, the supply voltage is varied based on throughput requirements. For higher throughput applications, the supply voltage is increased along with operating frequency, and vice versa for the lower throughput application. Sometimes this technique is also used to control power consumption and surface temperature. On-chip sensors sense temperature or current requirements and lower the supply voltage to reduce power consumption. Leakage-power mitigation can be achieved at the device level by applying multi-threshold voltage devices, multi-channel length devices, Stacking and Parking-states techniques. The following section gives details on these techniques.

### 3.3.3 Multiple-Threshold Voltages

Multiple-threshold voltages have been available on many, if not most, CMOS processes for a number of years. Since the standby power is so sensitive to the number of low-$V_T$ transistors, their usage, of the order of 5–10% of total transistors, is generally limited to fixing critical timing paths, especially in the physical design stage where other design changes have very long turnaround time. For instance, if the low $V_T$ is 110 mV less than the high $V_T$, 20% usage of the former will increase the chip standby power by nearly 500%. Low-$V_T$ insertion does not impact the active power component or design size. Obvious candidate circuits are SRAMs, whose power is dominated by leakage; higher $V_T$ generally also improves SRAM stability (as does a longer channel). The main drawbacks of low-$V_T$ transistors are that variation due to doping is uncorrelated between the high- and low-threshold transistors and that extra mask steps are needed, which incur additional process cost.

### 3.3.4 Long-Channel Transistors

The use of transistors that have longer than nominal channel length is another method of reducing leakage power. For example, by drawing a transistor 10 nm longer (long-*L*) than a minimum sized one, the Drain Induced Barrier Lowering (DIBL) is attenuated and the leakage can be reduced by 7 to 10 times on a 90 nm process. With this one change, nearly 20% of the total SRAM leakage component can be alleviated while maintaining performance. The loss in drive current due to increased channel resistance, of the order of 10–20%, can be made up for by an increase in width or, since the impact is to a single-gate stage, can be ignored for most of the designs [12]. The use of long-*L* is especially useful for SRAMs, since their overall performance is relatively insensitive to transistor delay. It can also be employed in other circuits if used judiciously. Compared with multiple threshold voltages, long-channel insertion has similar or lower process cost — it manifests as size increase rather than mask cost. It allows lower process complexity, and the different channel lengths track over process variation. It can be applied opportunistically to an existing design to limit leakage. A potential penalty is the increase in gate capacitance. Overall active power does not increase significantly if the activity factor of the affected gates is low, so this should also be considered when choosing target gates.

The target gate selection is driven by two main criteria. First, transistors must lie on paths with sufficient timing margin. Second, the highest leakage transistors should be chosen first from the selected

paths. The first criterion ensures that the performance goals are met. The second helps in maximizing leakage-power reduction. In order to use all of the available positive timing slack and avoid errors, long-*L* insertion is most advisable at the late design stages.

The long-*L* insertion can be done using standard cells designed using long-*L* transistors or by selecting individual transistors from the transistor-level design. Only the latter is applicable to full-custom design. There are advantages and disadvantages to both methods. For the cell level method, low-performance cells are designed with long-*L* transistors. For leakage reduction, high-performance cells on noncritical paths are replaced with lower performance cells with long-*L*. If the footprint and port locations are identical, then this method simplifies the physical convergence. Disadvantages of this method are that it requires a much larger cell library. It also requires a fine-tuned synthesis methodology to ensure long-*L* cell selection rather than lower performance nominal channel length cells. The transistor level flow has its own benefits. A unified flow can be used for custom- and auto-placed and routed blocks. Only a single nominal cell library is needed, albeit with space for long-*L* as mentioned.

### 3.3.5 Stacking and Parking States

There are two other design techniques for leakage-power reduction, Stacking and Parking states. In the Stacking technique, two or more transistors are placed in series. These transistor structures increase channel resistance of the leakage-current path in the OFF state. This technique gives ~2 to 3 times leakage-power reduction depending on the number of devices in the series. One needs to be careful about using this technique as it can increase switching power and gate leakage significantly. Stacking is beneficial in cases where a small number of transistors can add extra stack to a wide cone of logic or gate the power supply to it.

The main idea behind the Parking-states technique is to force the gates in the circuit to the low-leakage logic state when not in use [13]. There are three main pitfalls in this technique. First, additional logic is needed to generate the desirable state, which has area, switching power, as well as leakage-power cost. The second pitfall is that it is very specific to a design implementation. For example, a low-leakage state for a cone of logic may become high-leakage state if the logic implementation changes while keeping the same functionality. This technique is not advisable for random logic, but with careful implementation on structured data-path and memory arrays, it can save 2 to 5 times leakage power in the OFF state. The third pitfall is about making sure the design remains in the OFF state for long enough to make up for the dynamic power that is wasted during the forced transition to the low-leakage state.

Additional research on both of these techniques is needed with respect to the algorithmic aspects as well as the best way to incorporate them into synthesis flows.

### 3.3.6 Logic Styles

Dynamic circuits are generally regarded as more power dissipating than their static counterparts. While the power consumption of a static CMOS gates with constant inputs is limited to leakage power, dynamic gates may be continually precharging and discharging their output capacitance under certain input conditions.

For instance, if the inputs to the NAND gate in Figure 3.5(a) are stable, the output is stable. On the other hand, the dynamic NAND gate of Figure 3.5(b), under constant inputs A = B = 1, will keep rasing and lowering the output node, thus leading to a high level of energy consumption.

While this is true, it may not be such a relevant issue in high-performance designs, where most of the nodes will be switching most of the time. In fact, for several reasons, dynamic logic families are preferred in many high-speed, high-density designs (such as microprocessors). First, dynamic gates require fewer transistors, which means not only that they take up less space, but also that they exhibit a lower capacitance load, hence allowing for increased operation speed and for a reduction in the dynamic power dissipation component. Secondly, the evaluation of the output node can be performed solely through N-type MOSFET transistors, which further contributes to the improvement in performance. Thirdly, there is never a direct path from $V_{dd}$ to ground, thus totally eliminating the short-circuit power component. Finally, dynamic circuits intrinsically do not exhibit any hazards, potentially resulting in a significant

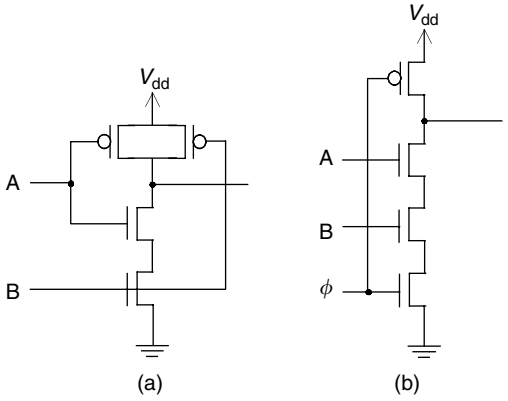


**FIGURE 3.5** NAND gate: (a) static CMOS; (b) dynamic domino.

reduction in power consumption. However, the design of dynamic circuits presents several issues that have been addressed through different design families [14].

Pass-transistor logic is another design style whose merits for low power, mainly due to the lower capacitance load of the input signal path, have been pointed out. The problem is that this design style may imply a significantly larger circuit.

## 3.4 Logic Synthesis for Low Power

A significant amount of CAD research has been carried out in the area of low power logic synthesis. By adding power consumption as a parameter for the synthesis tools, it is possible to save power with no, or minimal, delay penalty.

### 3.4.1 Logic Factorization

A primary means of technology-independent optimization is the factoring of logical expressions. For example, the expression $xy \vee xz \vee wy \vee wz$ can be factored into $(x \vee w)(y \vee z)$, reducing transistor count considerably. Common subexpressions can be found across multiple functions and reused. For area optimization, the kernels of the given expressions are generated and kernels that maximally reduce literal count are selected. Even though minimizing transistor count may in general reduce power consumption, in some cases the total switched capacitance actually increases. When targeting power dissipation, the cost function must take into account switching activity. The algorithms proposed for low power kernel extraction compute the switching activity associated with the selection of each kernel. Kernel selection then is based on the reduction of both area and switching activity [15].

### 3.4.2 Don't Care Optimization

Multilevel circuits are optimized by repeated two-level minimization with appropriate don't-care sets. The structure of the logic circuit may imply that some combinations on the inputs of a given logic gate never occur. These combinations form the controllability or satisfiability Don't care set (SDC) of the gate. Similarly, there may be some input combinations for which the output value of the gate is not used in the computation of any of the outputs of the circuit. The set of these combinations is called the observability Don't care set (ODC). Although initially don't-care sets were used for area minimization, techniques have been proposed for the use of don't-cares to reduce the switching activity at the output of a logic gate [16]. The transition probability of a static CMOS gate is given by $\alpha_x = 2p_x^0 p_x^1 = 2p_x^1(1 - p_x^1)$ (ignoring temporal correlation). The maximum for this function occurs for $p_x^1 = 0.5$. Therefore, in order to minimize the

switching activity, the strategy is to include don't care miniterms in the on-set of the function if $p_x^1 > 0.5$ or in the off-set if $p_x^1 < 0.5$.

### 3.4.3 Path Balancing

Spurious transitions account for a significant fraction of the switching activity power in typical combinational logic circuits. In order to reduce spurious switching activity, the delay of paths that converge at each gate in the circuit should be roughly equal, a problem known as path balancing. In the previous section, we discussed that transistor sizing can be tailored to minimize power basically at the cost of delaying signals not on the critical path. This approach has the additional feature of contributing to path balancing. Alternatively, path balancing can be achieved through the restructuring of the logic circuit, as illustrated in Figure 3.6.

### 3.4.4 Technology Mapping

Technology mapping is the process by which a logic circuit is implemented in terms of the logic elements available in a particular technology library. Associated with each logic element is the information about its area, delay, and internal and external capacitances. The optimization problem is to find the implementation that meets some delay constraint while minimizing cost that is a function of area and power consumption [17,18]. One of the main strategies to minimize power dissipation is to hide nodes with high switching activity within complex logic elements, as capacitances internal to gates are generally much smaller.

In many cases, the inputs of a logic gate are commutative in a Boolean sense. However, due to the way the gate is implemented, equivalent pins may present different input capacitance loads. In these cases, gate input assignment should be performed such that signals with high-switching activity map to the inputs that have lower input capacitance.

Additionally, most technology libraries include the same logic elements with different sizes (i.e., driving capability). Thus, in technology mapping for low power, the choice of the size of each logic element is made such that the delay constraints are met with minimum power consumption. This problem is the discrete counterpart of the transistor-sizing problem of the previous section.

### 3.4.5 State Encoding

The synthesis of sequential circuits offers new avenues of power optimization. State encoding is the process by which a unique binary code is assigned to each state in a finite-state machine (FSM). Although

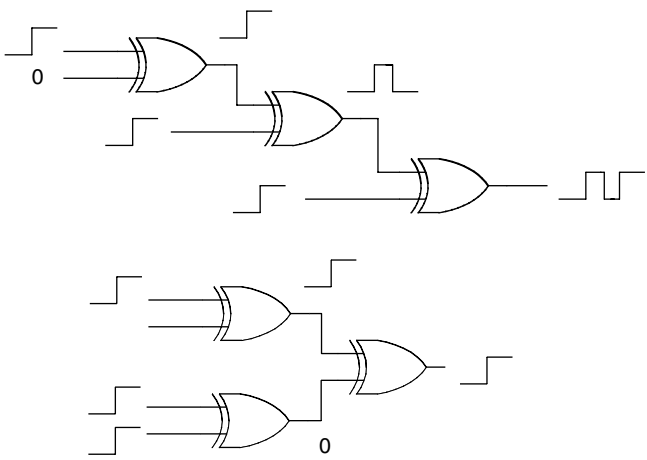


**FIGURE 3.6** Path balancing through logic restructuring to reduce spurious transitions.

this assignment does not influence the functionality of the FSM, it determines the complexity of the combinational logic block in the FSM implementation. State encoding for low power uses heuristics that assigns minimum Hamming distance codes to states that are connected by edges that have larger probability of being traversed [19]. The probability that a given edge in the state transition graph (STG) is traversed is given by the steady-state probability of the STG being in the start state of the edge, multiplied by the static probability of the input combination associated with that edge. Whenever this edge is exercised, only a small number of state lines (ideally one) will change, leading to reduced overall switching activity in the combinational logic block.

### 3.4.6 Finite-State Machine Decomposition

Finite-state machine decomposition has been proposed as a means for a low-power implementation of an FSM. The basic idea is to decompose the STG of the original FSM into two coupled STGs that together have the same functionality as the original FSM. Except for transitions that involve going from one state in one sub-FSM to a state in the other, only one of the sub-FSMs needs to be clocked. The strategy for state selection is such that only a small number is selected for one of the sub-FSMs. This selection consists in searching for a small cluster of states, such that the total probability of transitions between states in the cluster is high and the probability of transition to and from states outside the cluster is very low. The aim is to have a small sub-FSM that is active most of the time, disabling the larger sub-FSM. The reason for requiring a small number of transitions to and from the other sub-FSM is that this corresponds to the worst situation when both sub-FSMs are active. Each sub-FSM has an extra output that disables the state registers of the other sub-FSM, as shown in Figure 3.7. This extra output is also used to stop transitions at the inputs of the large sub-FSM. An approach to perform this decomposition solely using circuit techniques, thus without obtaining the STG, explicitly or implicitly, was proposed in [20].

Other techniques also based on blocking input signal propagation and clock-gating, such as pre-computation and guarded-evaluation, are covered in some detail in Chapter 13 of the first volume of this handbook.

### 3.4.7 Retiming

Retiming was first proposed as a technique to improve throughput by moving the registers in a circuit while maintaining input–output functionality. The use of retiming to minimize switching activity is based on the observation that the register outputs have significantly fewer transitions than the register inputs. In particular, no glitching is present. Moving registers across nodes through retiming may change the switching activity at several nodes in the circuit. In the circuit of Figure 3.8(a), the switched capacitance is given by $N_0C_B + N_1C_{FF} + N_2C_C$ and the switched capacitance in its retimed version, shown in

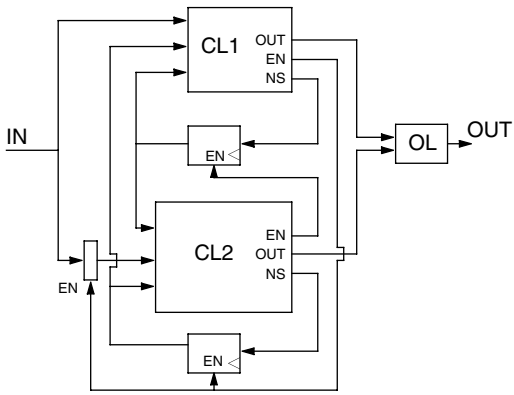


**FIGURE 3.7** Implementation diagram of a decomposed FSM for low power.

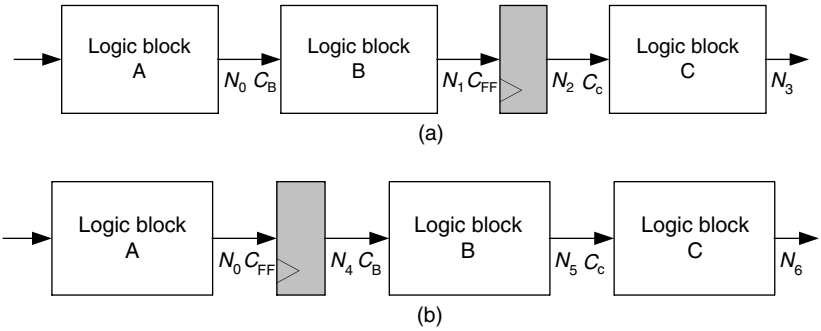


**FIGURE 3.8** Retiming for low power.

Figure 3.8(b), is $N_0C_{FF} + N_4C_B + N_5C_C$. One of these two circuits may have significantly less switched capacitance. Heuristics to place registers such that nodes driving large capacitances have reduced switching activity with a given throughput constraint have been proposed [21].

## 3.5 Conclusion

This chapter has covered methodologies for the reduction of power dissipation of digital circuits at the lower levels of design abstraction. The reduction of supply voltage has a large impact on power. However, it has the undesired consequence of reducing performance. Some of the techniques described apply local voltage reduction and dynamic voltage changes to minimize the impact of lost performance.

For most designs, the principal component of power consumption is related to the switching activity of the circuit during normal operation (dynamic power). The main strategy is to reduce the overall average switched capacitance, that is, the average amount of capacitance that is charged or discharged during circuit operation. The techniques presented address this issue by selectively reducing the switching activity of high-capacitive nodes, possibly at the expense of increasing this activity in other less capacitive nodes. Design automation tools using these approaches can save from 10 to 50% in power consumption with little area and delay overhead.

The static power component has been rising in importance with the reduction of feature size due to increased leakage and subthreshold currents. The most relevant methods, mostly at the circuit level, to minimize this power component have been presented.

Also covered in this chapter are power analysis tools. The power estimates provided can be used not only to indicate the absolute level of power consumption of the circuit, but also to direct the optimization process by indicating the most power efficient design alternatives.

## References

[1] N. Weste and K. Eshraghian, *Principles of CMOS VLSI Design: A Systems Perspective*, Addison-Wesley, Reading, MA, 1985.

[2] *The SPICE3 Implementation Guide*, University of California at Berkeley, ERL M89/44, 1989.

[3] PowerMill, Synopsys, http://www.synopsys.com/products/etg/powermill_ds.html.

[4] M. Johnson, D. Somasekhar, and K. Roy, Models and algorithms for bounds on leakage in CMOS circuits, *IEEE Trans. Comput.-Aided Design Integrated Circuits*, 18, 714–725, 1999.

[5] B. Sheu, D. Scharfetter, P. Ko, and M. Jeng, BSIM: Berkeley short-channel IGFET model for MOS transistors, *IEEE J. Solid-State Circuits*, 22, 558–566, 1987.

[6] J. Butts and G. Sohi, A static power model for architects, *Proceedings of MICRO-33*, Monterey, CA, 2000, pp. 191–201.

[7] W. Jiang, V. Tiwari, E. La Iglesia, and A. Sinha, Topological analysis for leakage prediction of digital circuits, *Proceedings of the International Conference on VLSI Design*, Bangalore, India, 2002, pp. 39–44.

[8] R. Burch, F. Najm, P. Yang, and T. Trick, A Monte Carlo approach to power estimation, *IEEE Trans. VLSI Systems*, 1, 63–71, 1993.

[9] A. Freitas and A.L. Oliveira, Implicit resolution of the Chapman–Kolmogorov equations for sequential circuits: an application in power estimation, in *Design, Automation and Test in Europe*, IEEE Computer Society, Los Alamitos, CA 2003, pp. 764–769.

[10] M. Pedram, Power minimization in IC design: principles and applications, *ACM Trans. Design Automation Electron. Syst.*, 1, 3–56, 1996.

[11] C-Y. Tsui, J. Monteiro, M. Pedram, S. Devadas, A. Despain, and B. Lin, Power estimation methods for sequential logic circuits, *IEEE Trans. VLSI Syst.*, 3, 404–416, 1995.

[12] L. Clark, R. Patel, and T. Beatty, Managing standby and active mode leakage power in deep submicron design, *International Symposium on Low Power Electronics and Design*, San Diego, CA, 2005.

[13] D. Lee and D. Blaauw, Static leakage reduction through simultaneous threshold voltage and state assignment, *Proceedings of the Design Automation Conference*, 2003, 2–6 June 2003, pp. 191–194.

[14] J. Yuan and C. Svensson, New single-clock CMOS latches and flipflops with improved speed and power savings, *IEEE J. Solid-State Circuits*, 32, 62–69, 1997.

[15] C-Y. Tsui, M. Pedram, and A. Despain, Power-efficient technology decomposition and mapping under an extended power consumption model, *IEEE Trans. CAD*, 13, 1110–1122, 1994.

[16] A. Shen, S. Devadas, A. Ghosh, and K. Keutzer, On average power dissipation and random pattern testability of combinational logic circuits, *Proceedings of the International Conference on Computer-Aided Design*, Santa Clara, CA, 1992, pp. 402–407.

[17] V. Tiwari, P. Ashar, and S. Malik, Technology mapping for low power in logic synthesis, *Integration, VLSI J.*, 20, pp.243–268, 1996.

[18] C. Tsui, M. Pedram, and A. Despain, Technology decomposition and mapping targeting low power dissipation, *Proceedings of the Design Automation Conference*, Dallas, USA, 1993, pp. 68–73.

[19] L. Benini and G. Micheli, State assignment for low power dissipation, *IEEE J. Solid-State Circuits*, 30, 258–268, 1995.

[20] J. Monteiro and A. Oliveira, Implicit FSM decomposition applied to low power design, *IEEE Trans. Very Large Scale Integration Syst.*, 10, 560–565, 2002.

[21] J. Monteiro, S. Devadas, and A. Ghosh, Retiming sequential circuits for low power, *Proceedings of the International Conference on Computer-Aided Design*, Santa Clara, CA, 1993, pp. 398–402.