Power Analysis and Optimization from Circuit to Register-Transfer Levels

José Monteiro, Rakesh Patel, and Vivek Tiwari

CONTENTS

3.1 Power Analysis 59
  3.1.1 Power Components in CMOS Circuits 59
  3.1.2 Analysis at the Circuit Level 61
  3.1.3 Static Power Estimation 62
  3.1.4 Logic-Level Power Estimation 63
    3.1.4.1 Simulation-Based Techniques 63
    3.1.4.2 Probabilistic Techniques 64
    3.1.4.3 Sequential Circuits 65
  3.1.5 Analysis at the Register-Transfer Level 66

3.2 Circuit-Level Power Optimization 66
  3.2.1 Transistor Sizing 67
  3.2.2 Voltage Scaling, Voltage Islands, and Variable $V_{dd}$ 67
  3.2.3 Multiple-Threshold Voltages 68
  3.2.4 Long-Channel Transistors 68
  3.2.5 Topological Techniques: Stacking and Parking States 69
  3.2.6 Logic Styles 71
The complexity and speed of today’s VLSI designs entail a level of power consumption that, if not addressed, causes an unbearable problem of heat dissipation. The operation of these circuits is only possible due to aggressive techniques for power reduction at different levels of design abstraction. The trends of mobile devices and the Internet of Things, on the other hand, drive the need for energy-efficient circuits and the requirement to maximize battery life. To meet these challenges, sophisticated design methodologies and algorithms for electronic design automation (EDA) have been developed.

One of the key features that led to the success of CMOS technology was its intrinsic low power consumption. It allowed circuit designers and EDA tools to concentrate on maximizing circuit performance and minimizing circuit area. Another interesting feature of CMOS technology is its nice scaling properties, which permitted a steady decrease in the feature size, allowing for numerous and exceptionally complex systems on a single chip, working at high 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. It also became apparent that the increasing integration of active elements per die area would lead to prohibitively large energy consumption of an integrated circuit. High power consumption is undesirable for economic and environmental reasons and also leads to high heat dissipation. In order to keep such a circuit working at acceptable temperature levels, expensive heat removal systems may be required.

In addition to the full-chip power consumption, and perhaps even more importantly, excessive heat is often dissipated at localized areas in the circuit, the so-called hot spots. This problem can be mitigated by selectively turning off unused sections of the circuit when such conditions are detected. The term dark silicon has been used to describe this situation where many available computational elements in an integrated circuit cannot be used at the same time [1]. These factors have contributed to the rise of power consumption as a major design parameter on par with performance and die size and a limitation of the continuing scaling of CMOS technology.

To respond to this challenge, intensive research has been invested in the past two decades in developing EDA tools for power optimization. Initial efforts focused on circuit- and logic-level tools, because at these levels EDA tools were more mature and malleable. Today, a large fraction of EDA research targets system- or architectural-level power optimization (Chapters 7 and 13 of Electronic Design Automation for IC System Design, Verification, and Testing, respectively), which promise 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 circuits’ 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 key CAD techniques proposed for low power design and synthesis. We start in Section 3.1 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 3.2 and 3.3, we review power optimization techniques at the circuit and logic levels of abstraction, respectively.

3.1 POWER ANALYSIS

Given the importance of power consumption in circuit design, EDA tools are required to provide power estimates for a circuit. When evaluating different designs, these estimates are needed to help identify the most power-efficient alternative. Since power estimates may be required for multiple alternatives, accuracy is sometimes sacrificed for tool response speed when the relative fidelity of estimates can be preserved. Second, an accurate 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 and delay estimates, because power depends not only on the circuit topology but also on the activity of the signals.

Typically, design exploration is performed at each level of abstraction, motivating power estimation tools at different levels. The higher the abstraction level, the less 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.

3.1.1 POWER COMPONENTS IN CMOS CIRCUITS

The power consumption of digital CMOS circuits is generally divided into three components [2]:

1. Dynamic power ($P_{dyn}$)
2. Short-circuit power ($P_{short}$)
3. Static power ($P_{static}$)

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

$$P = P_{dyn} + P_{short} + P_{static}$$

The dynamic power component, $P_{dyn}$, is related to the charging and discharging of the load capacitance at the gate output, $C_{out}$. 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 cut off and the pMOS transistor conducts. This creates a direct path between the voltage supply and $C_{out}$. Current $I_n$ 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, $E_c = \frac{1}{2}C_{out}V_{dd}^2$. 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 cut off 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 every time the output makes a transition. Given $N$ gate transitions within time $T$, its dynamic power consumption during that time period is given by

$$P_{dyn} = E_c \times N/T = \frac{1}{2} C_{out}V_{dd} \frac{N}{T}$$
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

$$P_{\text{dyn}} = E_c \times f_{clk} = \frac{1}{2} C_{\text{out}} V_{dd}^2 \alpha f_{clk}$$

$C_{\text{out}}$ is the sum of the three components $C_{\text{int}}, C_{\text{wire}},$ and $C_{\text{load}}$. Of these, $C_{\text{int}}$ represents the internal capacitance of the gate. This includes the diffusion capacitance of the drain regions connected to the output. $C_{\text{load}}$ represents the sum of gate capacitances of the transistors this logic gate is driving. $C_{\text{wire}}$ is 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 $\alpha C_{\text{out}}$ 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, $P_{\text{short}}$, is also related to the switching activity of the gate. During the transition of the input signal 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 that flows through both transistors during this period, indicated as $I_{\text{short}}$ in Figure 3.1. 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 limit this component to a small fraction of total power; hence, it is generally considered only a second-order effect. Given an estimate of the average amount of charge, $Q_{\text{short}}$, that is carried by the short-circuit current per output transition, the short-circuit power is obtained by

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

The static power component, $P_{\text{static}}$, 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 is usually 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, that is, 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 $V_{GS} - V_T$. With the reduction of transistor size, leakage current tends to increase for each new technology node, driving up the relative weight of static power consumption. This problem has been mitigated through the introduction of high-$\kappa$ dielectric materials and new gate geometry architectures [3].
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 should be avoided in practice.

This condition 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 precharged 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 exhibits a constant current flowing whenever the output is at logic 0, that is, when there is a direct path to ground through the nMOS network.

### 3.1.2 Analysis at the Circuit Level

Power estimates at the circuit level are generally obtained using a circuit-level simulator, such as SPICE [4]. 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 all power sources).

At this level, complex models for the circuit devices can be used. These models permit the accurate computation of the three components of power—dynamic, short-circuit, 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. After placement and routing of the circuit, simulation can handle back-annotated circuit descriptions, that is, with realistic interconnect capacitive and resistive values. The power estimates thus obtained can be very close to the power consumption of the actual fabricated circuit.

The problem is that such detailed simulation requires the solution of complex systems of equations and is only practical for small circuits. Another limitation is that the input sequences must necessarily be very short since simulation is time consuming; hence, the resulting power estimates may poorly reflect the real statistics of the inputs. For these reasons, full-fledged circuit-level power estimation is typically only performed for the accurate characterization of small-circuit modules. To apply circuit-level simulation to larger designs, one can resort to very simple models for the active devices. Naturally, this simplification implies accuracy loss. On the other hand, massively parallel computers extend the applicability of these methods to even larger designs [5].

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 simplified model allows 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. At early technology nodes, designers were willing to ignore this power component since it accounted for a negligible fraction of total power, but now its relative importance is increasing. Leakage power estimation must then be performed independently using specifically tailored tools. Many different approaches have been proposed, some of which are presented in the next section.

Among intermediate-complexity solutions, PrimeTime PX, an add-on to Synopsys’ static timing analysis tool [6], offers power estimates with accuracy close to SPICE. This tool employs table lookup of current models for given transistor sizes and uses circuit partitioning to solve the circuit equations independently on each partition. Although some error is introduced by not accounting for interactions between different partitions, this technique greatly simplifies the problem to be solved, allowing for fast circuit-level estimates of large designs.
3.1.3 STATIC POWER ESTIMATION

Static power analysis is typically performed using the subthreshold model to estimate leakage per unit micron, which is then extrapolated 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 [7]. Analysis can be viewed as the modification of this total width by the stacking factor.

Most analytical works on leakage have used the BSIM2 subthreshold current model [8]:

\[
I_{sub} = A e^{\left(\frac{V_{GS} - V_T - \gamma V_{SB} + \eta V_{DS}}{nV_{TH}}\right)} \left(1 - e^{\frac{-V_{GS}}{V_{TH}}}\right)
\]

where

- \(V_{GS}\), \(V_{DS}\), and \(V_{SB}\) are the gate-source, drain-source, and source-bulk voltages, respectively
- \(V_T\) is the zero-bias threshold voltage
- \(V_{TH}\) is the thermal voltage \((kT/q)\)
- \(\gamma'\) is the linearized body-effect coefficient
- \(\eta\) is the drain-induced barrier lowering (DIBL) coefficient

A = \(\mu_0C_{ox}(W/L_{off})W_{TH}^2e^{1.8}\)

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 [9,10]. For example, a simple empirical model is as follows [10]:

\[
I_{leak} = I_{off} \frac{W_{tot}}{X_s} X_t
\]

where

- \(I_{off}\) is the leakage current per micron of a single transistor measured from actual silicon at a given temperature
- \(W_{tot}\) is the total transistor width (sum of all N and P devices)
- \(X_s\) is an empirical stacking factor based on the observation that transistor stacks leak less than single devices
- \(X_t\) is the temperature factor and is used to scale \(I_{off}\) to the appropriate junction temperature of interest

The \(I_{off}\) value is typically specified at room temperature (therefore the need for a temperature factor to translate to the temperature of interest).

The other major component of static power is gate leakage. Gate leakage effectively becomes a first-order effect only when the gate oxide is thin enough such that direct quantum tunneling through the oxide becomes a significant quantity. The amount of gate leakage current is directly proportional to transistor width, and thus, the main effort for gate leakage estimation is to estimate the total transistor width, \(W_{tot}\), similar to what is required for subthreshold current. The exact value of gate leakage depends on the gate-to-source and drain-to-source voltages, \(V_{GS}\) and \(V_{DS}\), and these depend on gate input values. State-based weighting of the transistor width can therefore be used for more accurate gate leakage estimation. However, this entails the additional effort of estimating the state probabilities for each gate [11].

© 2016 by Taylor & Francis Group, LLC
At present, $V_T$ is high enough such that subthreshold current is dominated by the dynamic component of the total active current. On the other hand, subthreshold current dominates the total standby current when compared to gate and well leakage components. As oxide thickness continued to scale down, it was feared that gate leakage would become the dominant source of leakage. However, the introduction of new technologies like metal gates and 3D FinFETs has decelerated the trend toward thinner oxides, and therefore, subthreshold leakage will continue to dominate gate leakage for at least a couple of more technology nodes.

### 3.1.4 LOGIC-LEVEL POWER ESTIMATION

A key observation from Section 3.1.1 that facilitates power estimation at the logic level 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, $C_{out}$, the gate is driving, $E_c = \frac{1}{2}C_{out}V_{dd}^2$. Given parasitic gate and wire capacitance models that allow the computation of $C_{out,i}$ 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, that is, the switching activity of the gate. This corresponds to either parameter $N$ or $\alpha$, and we need to only apply Equation 3.2 or 3.3, respectively, to obtain power.

Naturally, this estimate refers only to the dynamic power component. For total power consumption, we 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, and hence, only an estimate for this component is required. There are two classes of techniques for the switching activity computation, namely, simulation-based and probabilistic analyses (also known as dynamic and static techniques, respectively).

#### 3.1.4.1 SIMULATION-BASED TECHNIQUES

In simulation-based switching activity estimation, highly optimized logic simulators are used, allowing for 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 and wires, meaning that all transitions in the circuit occur at the same time instant. Hence, each gate makes at most one transition per input vector. In reality, logic gates have 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, implying 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, $w$ switches to 0, thus 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 reconvergent signals, such as multipliers, may be more than 50% [12]. 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

![Figure 3.2 Example of a logic circuit with glitching and spatial correlation.](image-url)
delay model where gate delays are retrieved from a precharacterized library of gates. Process variation introduces another level of complexity, motivating a statistical analysis for delay, and consequently of the spurious activity [13].

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 only input statistics are given, a 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 [14] uses the central limit theorem to determine the number of input vectors that must be simulated:

$$N \geq \left( \frac{z_{\theta/2}\delta}{\pi\varepsilon} \right)^2$$

where
- $N$ is the number of input vectors
- $\mathcal{P}$ and $s$ are the measured average and standard deviation of the power
- $z_{\theta/2}$ is obtained from the normal distribution

In practice, for typical combinational circuits and reasonable error and confidence levels, the number of input vectors needed to obtain the overall average switching activity is typically very small (thousands) even for complex logic circuits. However, in many situations, accurate average switching activity for each node 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 for these nodes, based on the argument that these are the nodes that have less impact on the dynamic power consumption of the circuit.

Still, today’s highly parallel architectures facilitate fast simulation of a large number of input vectors, thus improving the accuracy of this type of Monte Carlo–based estimation methods [15].

### 3.1.4.2 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 associated with 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 probabilities 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 mentioned earlier. 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 the transport delays to be $\Delta_1$ and $\Delta_2$ for the gates in the circuit of Figure 3.2 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 are analyzed 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
probabilities of the signals being at 1 is 0.5. However, it is not true that $p_x = 1 - p_w p_y = 0.625$ because signals $w$ and $y$ are not independent: $p_x p_y = (1 - p_w p_y) p_y = p_y - p_y p_y$ (note that $p_w p_y = p_y$), giving $p_x = (1 - p_x + p_x p_y) = 0.75$. 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 [16], but it cannot be applied to very large designs due to its high complexity.

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, and 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, and hence, the probability of the signals being at 1 is $p_x^i = p_y^i = 0.5$ (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 $\alpha = p_x^{0i} + p_y^{10} = p_x^1 p_y^i + p_y^i p_x^0 = 0.5$. However, we can see that, during the depicted time interval, signal $x$ remains low for three clock cycles, remains 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 $p_x^{0i} = \frac{4}{8} = 0.375$, $p_x^{1i} = \frac{4}{8} = 0.125$, $p_y^{10} = \frac{4}{8} = 0.125$, and $p_y^{01} = \frac{3}{8} = 0.375$. Therefore, the actual average switching activity of $x$ is $\alpha_x = p_x^{0i} + p_x^{1i} = 0.25$. As for signal $y$, it never remains low or high, making a transition on every clock cycle. Hence, $p_y^{0i} = p_y^{10} = 0$ and $p_y^{01} = p_y^{10} = \frac{1}{2} = 0.5$, and the actual average switching activity of $y$ is $\alpha_y = p_y^{0i} + p_y^{10} = 1.0$. 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 exact modeling of these issues makes the computation of the average switching activity an NP-hard problem, meaning that exact methods are only applicable to small circuits and thus are of little practical interest. Many different approximation schemes have been proposed [17].

### 3.1.4.3 SEQUENTIAL CIRCUITS

Computing the switching activity for sequential circuits is significantly more difficult, because the state space must be visited in a representative manner to ensure the accuracy of the state signal probabilities. 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 signals 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 of the circuit is not possible, since these are exponential in the number of sequential elements in the circuit. A common approximation is to compute the transition activity.
probabilities for each state signal [18]. To partially recover the spatial correlation between state signals, a typical approach is to duplicate the subset of logic that generates the next state signals and append it to the present state signals, as is illustrated in Figure 3.4. Then the method for combinational circuits is applied to this modified network, ignoring the switching activity in the duplicated next state logic block.

3.1.5 ANALYSIS AT THE REGISTER-TRANSFER LEVEL

At the register-transfer level (RTL), the circuit is described in terms of interconnected modules of varied complexity from simple logic gates to full-blown multipliers. Power estimation at this level determines the signal statistics at the input of each of these modules and then feeds these values to the module’s power model to evaluate its power dissipation. These models are normally available with the library of modules. One way to obtain these power models is to characterize the module using logic- or circuit-level estimators, a process known as macromodeling. We refer to Chapter 13 of Electronic Design Automation for IC System Design, Verification, and Testing, where this topic is discussed in more detail.

3.2 CIRCUIT-LEVEL POWER OPTIMIZATION

From the equations that model power consumption, one sees that a reduction of the supply voltage has the largest impact on power reduction, given its quadratic effect. This has been the largest source of power reductions and is widely applied across the semiconductor 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.

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 it with higher switching in nodes with lower capacitance.

Static power due to leakage current, however, presents a different set of challenges. As Equation 3.5 shows, reducing $V_{dd}$ will reduce leakage current as well. However, reducing the number of transitions to reduce switched capacitance has little benefit, since leakage power is consumed whether or not there is a transition in the output of a gate. The most effective way to reduce leakage is to effectively shut off the power to a circuit—this is called power gating. Other techniques are motivated by the relationship of leakage current to the threshold voltage $V_T$. Increasing the threshold voltage reduces the leakage current. Equation 3.6 motivates other techniques that exploit the relationship of leakage current to circuit topology.
In the following, we briefly discuss the key circuit-level techniques that have been developed to address each of these points. There is a vast amount of published work that covers these and other techniques in great detail, and the interested reader is recommended to start with books [19] and overview papers [20] that cover these topics in greater depth.

3.2.1 TRANSISTOR SIZING

The propagation delay (usually just referred to as delay) of a gate is dependent on the gate output resistance and the total capacitance (interconnect and load) [2]. Transistor sizing (or gate sizing) helps reduce delay by increasing gate strength at the cost of increased area and power consumption. Conversely by reducing gate strength, the switched capacitance, and therefore, the power, can be reduced at the cost of increased delay. This trade-off can be performed manually for custom designs or through the use of automated tools.

Up until the recent past, large parts of high-performance CPUs were typically custom designed. Even now, the most performance critical parts of high-performance designs have a mix of synthesized and custom designed parts. Such designs may involve manual tweaking of transistors to upsize drivers along 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 overdesign that is wasteful of power. An emphasis on lower power, however, motivates the identification of such sources of power waste. An example of such waste is circuit paths that are designed faster than they 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 gains over manual design methodologies.

The use of multiple-threshold voltages (“multi-$V_T$”) to reduce leakage power in conjunction with traditional transistor sizing is now a widely used design technique. The main idea here is to use lower-$V_T$ transistors in critical paths rather than large high-$V_T$ transistors. However, this technique increases subthreshold leakage due to low $V_T$. So, it is very important to use low-$V_T$ transistor selectively and optimize their usage to achieve a good balance between capacitive current and leakage current in order to minimize the total current. This consideration is now part of the postsynthesis or postlayout automated tools and flows that recognize both low-$V_T$ and high-$V_T$ substitution. For example, after postlayout timing analysis, a layout tool can operate in incremental mode to do two things: insert low-$V_T$ cells into critical paths to improve speed and insert higher-$V_T$ cells into noncritical paths to bring leakage back down again.

Custom designers may have the flexibility to manually choose the transistor parameters to generate custom cells. Most synthesized designs, however, only have the choice of picking from different gates or cells in a cell library. These libraries typically have a selection of cells ranging from high performance (high power) to low power (low performance). In this case, the transistor-sizing problem reduces to the problem of optimal cell selection either during the initial synthesis flow or of tweaking the initial selection in a postsynthesis flow. This has been an area of active academic research [21] as well as a key optimization option in commercial tools [19].

3.2.2 VOLTAGE SCALING, VOLTAGE ISLANDS, AND VARIABLE $V_{DD}$

As mentioned earlier, the reduction of $V_{dd}$ is the most effective way of 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 also needs to be scaled down since circuit speed is roughly inversely proportional to $(V_{dd} - V_T)$. Typically, $V_{dd}$ should be larger than 4$V_T$ if speed is not to suffer excessively. As the threshold voltage decreases, subthreshold leakage current increases exponentially. With every 0.1 V reduction in $V_T$, subthreshold current increases by 10 times.
In the nanometer technologies, with further $V_T$ reduction, subthreshold current has become a significant portion of the overall chip current. At 0.18 m feature size and less, leakage power starts eating into the benefits of lower $V_{dd}$. In addition, the design of dynamic circuits, caches, sense amps, PLAs, etc., becomes difficult at higher subthreshold leakage currents. Lower $V_{dd}$ also exacerbates noise and reliability concerns. To combat the subthreshold current increase, various techniques have been developed, as mentioned in the Section 3.2.5.

Voltage islands and variable $V_{dd}$ are variations of voltage scaling that can be used at the circuit level. Voltage scaling is mainly technology dependent and typically applied to the whole chip. Voltage islands are more suitable for system-on-chip design, which integrates different functional modules with various performance requirements onto a single chip. We refer to the chapter on RTL power analysis and optimization techniques for more details on voltage islands. The variable voltage and voltage island techniques are complementary and can be implemented on the same block to be 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 measure 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 multithreshold voltage devices, multichannel length devices, and stacking and parking state techniques. The following section gives details on these techniques.

### 3.2.3 MULTIPLE-THRESHOLD VOLTAGES

Multiple-threshold voltages (most often a high-$V_T$ and a low-$V_T$ option) have been available on many, if not most, CMOS processes for a number of years. For any given circuit block, the designer may choose to use one or the other $V_T$ or a mixture of the two. For example, use high-$V_T$ transistor as the default and then selectively insert low-$V_T$ transistors. Since the standby power is so sensitive to the number of low-$V_T$ transistors, their usage, in the order of 5%–10% of the total number of transistors, is generally limited to only fixing critical timing paths, or else leakage power could increase dramatically. For instance, if the low-$V_T$ value is 110 mV less than the high-$V_T$ value, 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, and it is often the easiest option in the postlayout stage, leading to the least layout perturbation. Obvious candidate circuits for using high-$V_T$ transistors as the default and only using selectively low-$V_T$ transistors are SRAMs, whose power is dominated by leakage, and a higher $V_T$ generally also improves SRAM stability (as does a longer channel). The main drawbacks of low-$V_T$ transistors are that delay variations due to doping are uncorrelated between the high- and low-threshold transistors, thus requiring larger timing margins, and that extra mask steps are needed, which incur additional process cost.

### 3.2.4 LONG-CHANNEL TRANSISTORS

The use of transistors that have longer than nominal channel length is another method of reducing leakage power [22]. For example, by drawing a transistor 10 nm longer (long-$L$) than a minimum sized one, the DIBL is attenuated and the leakage can be reduced by $7\times-10\times$ on a 90 nm process. With this one change, nearly 20% of the total SRAM leakage component can be eliminated while maintaining performance. The loss in drive current due to increased channel resistance, on the order of 10%–20%, can be compensated by an increase in width or since the impact is on a single gate stage, it can be ignored for most of the designs [22]. The use of long-$L$ is especially useful for SRAMs, since their overall performance is relatively insensitive to transistor delay. It can also be applied to other circuits, if used judiciously. Compared with multiple-threshold voltages, long-channel insertion has similar or lower process cost—it manifests as size increases 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 criterion 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 performed by 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. Unfortunately, this method 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 blocks and auto placed-and-routed blocks. Only a single nominal cell library is needed, albeit with space for long-$L$ as mentioned.

3.2.5 TOPOLOGICAL TECHNIQUES: STACKING AND PARKING STATES

Another class of techniques exploits the dependence of leakage power on the topology of logic gates. Two examples of such techniques are stacking and parking states. These techniques are based on the fact that a stack of “OFF” transistors leaks less than when only a single device in a stack is OFF. This is primarily due to the self-reverse biasing of the gate-to-source voltage $V_{GS}$ in the OFF transistors in the stack. Figure 3.5 illustrates the voltage allocation of four transistors in series [10]. As one can see, $V_{GS}$ is more negative when a transistor is closer to the top of the stack. The transistor with the most negative $V_{GS}$ is the limiter for the leakage of the stack. In addition, the threshold voltages for the top three transistors are increased because of the reverse-biased body-to-source voltage (body effect).

Both the self-reverse biasing and the body effects reduce leakage exponentially as shown in Equation 3.5. Finally, the overall leakage is also modulated by the DIBL effect for submicron MOSFETs. As $V_{DS}$ increases, the channel energy barrier between the source and the drain is lowered. Therefore, leakage current increases exponentially with $V_{DS}$.

The combination of these three effects results in a progressively reduced $V_{DS}$ distribution from the top to the bottom of the stack, since all of the transistors in series must have the same leakage current. As a result, significantly reduced $V_{DS}$, the effective leakage of stacked transistors, is much lower than that of a single transistor.

Table 3.1 quantifies the basic characteristics of the subthreshold leakage current for a fully static four-input NAND gate. The minimum leakage condition occurs for the “0000” input vector (i.e., all inputs a, b, c, and d are at logic zero). In this case, all the PMOS devices are “ON” and

![Figure 3.5](image-url)

© 2016 by Taylor & Francis Group, LLC
the leakage path exists between the output node and the ground through a stack of four NMOS devices. The maximum leakage current occurs for the “1111” input case, when all the NMOS devices are ON and the leakage path, consisting of four parallel PMOS devices, exists between the supply and the output node. The stacking factor variation between the minimum and maximum leakage conditions reflects the magnitude of leakage dependence on the input vector. In the four-input NAND case, we can conclude that the leakage variation between the minimum and maximum cases is a factor of about 40 (see Table 3.1). The values were measured using an accurate SPICE-like circuit simulator on a 0.18 μm technology library. The average leakage current was computed based on the assumption that all the 16 input vectors were equally probable.

Stacking techniques take advantage of the effects described earlier to increase the stack depth [23]. One of the examples is the sleep transistor technique. This technique inserts an extra series-connected device in the stack and turns it OFF during the cycles when the stack will be OFF as a whole. This comes at the cost of the extra logic to detect the OFF state, as well as the extra delay, area, and dynamic power cost of the extra device. Therefore, this technique is typically applied at a much higher level of granularity, using a sleep transistor that is shared across a larger block of logic. Most practical applications in fact apply this technique at a very high level of granularity, where the sleep state (i.e., inactive state) of large circuit blocks such as memory and ALUs can be easily determined. At that level, this technique can be viewed as analogous to power gating, since it isolates the circuit block from the power rails when the circuit output is not needed, that is, inactive. Power gating is a very effective and increasingly popular technique for leakage reduction, and it is supported by commercial EDA tools [24], but it is mostly applied at the microarchitectural or architectural level and therefore not discussed further in here.

The main idea behind the parking state technique is to force the gates in the circuit to the low-leakage logic state when not in use [25]. As described earlier, leakage current is highly dependent on the topological relationship between ON and OFF transistors in a stack, and thus, leakage depends on the input values. This technique avoids the overhead of extra stacking devices, but additional logic is needed to generate the desirable state, which has an area and switching power cost. This technique is not advisable for random logic, but with careful implementation for structured datapath and memory arrays, it can save significant leakage power in the OFF state.

One needs to be careful about using these techniques, given the area and switching overheads of the introduced devices. Stacking is beneficial in cases where a small number of transistors can add extra stack length to a wide cone of logic or gate the power supply to it. The delays introduced by the sleep transistors or by power gating also imply that these techniques are beneficial only when the targeted circuit blocks remain in the OFF state for long enough to make up for the overhead of driving the transitions in and out of the OFF states. These limitations can be overcome with careful manual intervention or by appropriate design intent hints to automated tools.

Leakage power reduction will remain an active area of research, since leakage power is essentially what limits the reduction of dynamic power through voltage scaling. As transistor technology scales down to smaller feature sizes, making it possible to integrate greater numbers of devices on the same chip, additional advances in materials and transistor designs can be expected to allow for finer-grained control on the power (dynamic and leakage) and performance trade-offs. This will need to be coupled with advances in power analysis to understand nanometer-scale effects that have so far not been significant enough to warrant detailed power models. In conjunction with these models, new circuit techniques to address these effects will need to be developed. As these circuit techniques gain wider acceptability and applicability, algorithmic research to incorporate these techniques in automated synthesis flows will continue.

| TABLE 3.1 | Stacking Factors of Four-Input NAND |
|---|---|---|---|
| Stacking factor Xs | Minimum | Maximum | Average |
| Input vector (a b c d) | 1.75 | 70.02 | 9.95 |
| (1 1 1 1) | (0 0 0 0) | — |

© 2016 by Taylor & Francis Group, LLC
3.2.6 LOGIC STYLES

Dynamic circuits are generally regarded as dissipating more power than their static counterparts. While the power consumption of a static CMOS gate 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 3.6a are stable, the output is stable. On the other hand, the dynamic NAND gate of Figure 3.6b, under constant inputs $A = B = 1$, will keep raising and lowering the output node, thus leading to high energy consumption.

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 capacitive load, hence allowing for increased operation speed and for reduced dynamic power dissipation. Second, the evaluation of the output node can be performed solely through N-type MOSFET transistors, which further contributes to the improvement in performance. Third, there is never a direct path from $V_{dd}$ to ground, thus effectively eliminating the short-circuit power component. Finally, dynamic circuits intrinsically do not create any spurious activity, which can make for a significant reduction in power consumption. However, the design of dynamic circuits presents several issues that have been addressed through different design families [26].

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

Sequential circuit elements are of particular interest with respect to their chosen logic style, given their contribution to the power consumption of a logic chip. These storage elements—flip-flops or latches—are the end points of the clock network and constitute the biggest portion of the switched capacitance of a chip because of both the rate at which their inputs switch (every clock edge) and their total number (especially in high-speed circuits with shallow pipeline depths). For this reason, these storage elements received a lot of attention [27]. For example, dual-edge-triggered flip-flops have been proposed as a lower-power alternative to the traditional single-edge-triggered flip-flops, since they provide an opportunity to reduce the effective clock frequency by half. The trade-offs between ease of design, design portability, scalability, robustness, and noise sensitivity, not to mention the basics trade-offs of area and performance, require these choices to be made only after a careful consideration of the particular design application. These trade-offs also vary with technology node, as the leakage power consumption must be factored into the choice.

In general, one can expect research and innovation in circuit styles to continue as long as the fundamental circuit design techniques evolve to overcome the limitations or exploit the opportunities provided by technology scaling.

**FIGURE 3.6** NAND gate: (a) static CMOS and (b) dynamic domino.

© 2016 by Taylor & Francis Group, LLC
3.3 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.

3.3.1 LOGIC FACTORIZATION

A primary means of technology-independent optimization is the factoring of logical expressions. For example, the expression \(xy \lor xz \lor wy \lor wz\) can be factored into \((x \lor w)(y \lor z)\), reducing transistor count considerably. Common subexpressions can be found across multiple functions and reused. For area optimization, several candidate divisors (e.g., kernels) of the given expressions are generated and those that maximally reduce literal count are selected. Even though minimizing transistor count may, in general, reduce power consumption, in some cases the total effective 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 is based on the reduction of both area and switching activity [28].

3.3.2 DON’T-CARE OPTIMIZATION

Multilevel circuits are optimized taking into account appropriate don’t-care sets. The structure of the logic circuit may imply that some input combinations of a given logic gate never occur. These combinations form the controllability or satisfiability don’t-care set 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. Although initially don’t-care sets were used for area minimization, techniques have been proposed for the use of don’t-care sets to reduce the switching activity at the output of a logic gate [29]. The transition probability of a static CMOS gate is given by \(\alpha_s = 2p_t^3p_i^3 = 2p_t^3(1 - p_i^3)\) (ignoring temporal correlation). The maximum for this function occurs for \(p_i^3 = 0.5\). Therefore, in order to minimize the switching activity, the strategy is to include minterms from the don’t-care set in the onset of the function if \(p_i^3 > 0.5\) or in the off-set if \(p_i^3 < 0.5\).

3.3.3 PATH BALANCING

Spurious transitions account for a significant fraction of the switching activity power in typical combinational logic circuits [30]. 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 primarily 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 3.7.

![FIGURE 3.7 Path balancing through logic restructuring to reduce spurious transitions.](image-url)
Path balancing is extremely sensitive to propagation delays, becoming a more difficult problem when process variations are considered. The work in [30] addresses path balancing through a statistical approach for delay and spurious activity estimation.

3.3.4 TECHNOLOGY MAPPING

Technology mapping is the process by which a logic circuit is realized 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 the delay constraint while minimizing a cost function that is a function of area and power consumption [31,32]. To minimize power dissipation, nodes with high switching activity are mapped to internal nodes of complex logic elements, as capacitances internal to gates are generally much smaller.

In many cases, the inputs of a logic gate are commutative in the Boolean sense. However, in a particular gate implementation, 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 size of each logic element is chosen so that the delay constraints are met with minimum power consumption. This problem is the discrete counterpart of the transistor-sizing problem described in the previous section.

3.3.5 STATE ENCODING

The synthesis of sequential circuits offers new avenues for 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 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 assign minimum Hamming distance codes to states that are connected by edges that have larger probability of being traversed [33]. 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 signals (ideally one) will change, leading to reduced overall switching activity in the combinational logic block.

3.3.6 FSM DECOMPOSITION

FSM decomposition has been proposed for 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 of states is selected for one of the sub-FSMs. This selection consists of searching for a small cluster of states such that summation of the probabilities of transitions between states in the cluster is high, and there is a very low probability of transition to and from states outside of the cluster. The aim is to have a small sub-FSM that is active most of the time, disabling the larger sub-FSM. Having a small number of transitions to/from the other sub-FSM corresponds to the worst case, 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 3.8. 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 any derivation of the STG, was proposed in [34].
Other techniques based on blocking input signal propagation and clock gating, such as pre-computation, are covered in some detail in Chapter 13 of *Electronic Design Automation for IC System Design, Verification, and Testing*.

### 3.3.7 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 output of a register has significantly fewer transitions than its input. 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 shown in Figure 3.9a, the switched capacitance is given by \( N_0 C_B + N_1 C_{FF} + N_2 C_C \), and the switched capacitance in its retimed version, shown in Figure 3.9b, is \( N_6 C_{FF} + N_4 C_B + N_5 C_C \). One of these two circuits may have significantly less switched capacitance. Heuristics to place registers such that nodes driving large capacitances have a reduced switching activity, subject to a given throughput constraint, have been proposed [35].

### 3.4 SUMMARY

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 also reduces performance. Some of the techniques we described apply local voltage reduction and dynamic voltage control 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 here 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 we presented address this issue by selectively reducing the switching activity of high-capacitance nodes, possibly at the expense of increasing the activity of other less capacitive nodes. Design automation tools using these approaches can save 10%–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. Key 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

5. L. Han, X. Zhao, and Z. Feng, TinySPICE: A parallel SPICE simulator on GPU for massively repeated small circuit simulations, *Design Automation Conference (DAC)*, Austin, TX, 2013, pp. 89:1–89:8.


