Chapter 2
Power Dissipation

Wei Zhang, James Williamson, and Li Shang

Abstract This chapter discusses the power consumption issue of the mainstream CMOS technologies. During the past two decades, power dissipation has stood out as the foremost design challenge for general-purpose and application-specific integrated circuits (ICs). Considering and optimizing the circuit power efficiency has become essential. IC power modeling, analysis, design-time optimization, and run-time management techniques have been intensively studied. This chapter covers the basics of the IC power consumption issue. It first investigates the sources of IC power dissipation, and then discusses recent techniques for IC power analysis. Finally, it studies recently proposed power optimization techniques from circuit and physical design to system synthesis.

2.1 Introduction

Technology trends and design constraints drive the evolution of integrated circuit (IC) and system design. For decades, electronic systems have benefited from the relentless progress of semiconductor fabrication technology governed by Moore's Law. With each technology generation, ICs become denser, faster, and cheaper. However, with increasing technology scaling and system integration, IC power consumption has become a major design challenge, introducing a variety of power-related design issues.

Low-power IC design has been an active research field over the past 20 years. From modeling and analysis to design-time optimization and run-time management, IC power consumption issues have been thoroughly studied and carefully optimized. The existing large body of low-power IC research has effectively alleviated the IC power consumption challenges, and the semiconductor technology is thus allowed

L. Shang (✉)
Department of Electrical, Computer, and Energy Engineering, University of Colorado, Boulder, Co, USA
e-mail: li.shang@colorado.edu

to continue to scale. For instance, from Intel Pentium 4 uni-core microprocessor to Intel Core i7 quad-core design, the microprocessor thermal design power remains approximately the same, but the overall system performance has been improved substantially. However, IC power consumption challenges will continue to grow. As projected by International Technology Roadmap for Semiconductors (ITRS) [1], power will remain to be a limiting factor in future technologies. The power optimization techniques, which have been widely deployed in the past, such as voltage frequency scaling, clock gating, and power gating, become less effective and applicable as technical scales. For instance, Intel technology blueprint shows that from 32 nm and beyond, with each technology generation, voltage and frequency scaling is tightly constrained, and the corresponding power benefits become marginal. On the other hand, this is not the first time we face the power consumption issue. Indeed, power consumption has been a recurring challenge for electronic system design. As shown in Fig. 2.1, power efficiency was one of the main motivations behind the technology transition from vacuum tube to bipolar, and then to CMOS during the past several decades. Device innovations have been the most effective solution. In addition, the historical trend implies that we will soon face another technology transition. However, CMOS technology has entered the nanometer regime. Further technology scaling becomes increasingly challenging. Emerging nano devices, such as carbon nanotubes, nanowires, and graphene, have demonstrated potential, but still face major challenges for large-scale chip and system integration. To build future

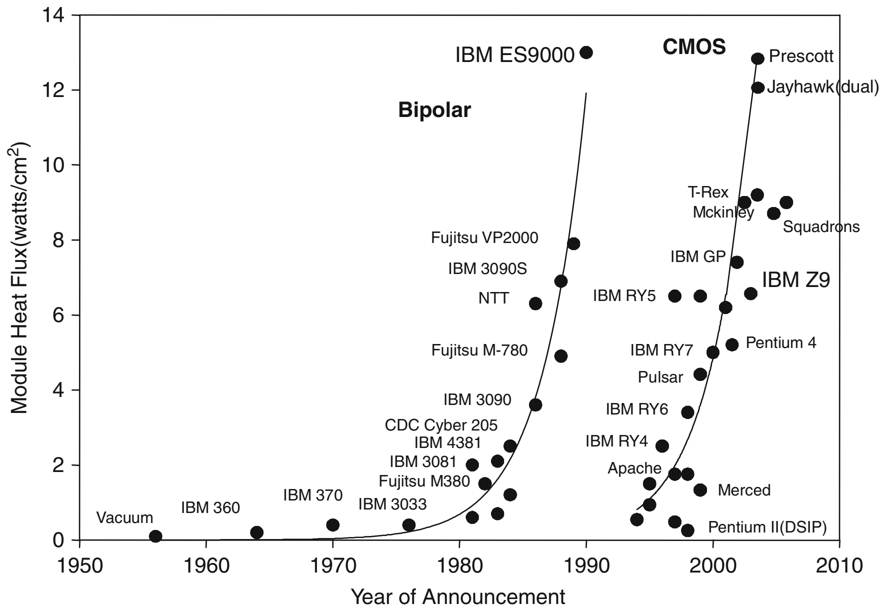


Fig. 2.1 Power consumption: a recurrent challenge

power-efficient systems, we need systematic innovations from system integration to device and fabrication technologies.

The rest of this chapter covers the basics of the IC power consumption issue. It first investigates the sources of IC power dissipation, and then discusses recent techniques for IC power modeling and analysis. Finally, it studies recently proposed power optimization techniques from circuit and physical design to system synthesis.

2.2 Source of Power Dissipation in CMOS Digital Circuits

In CMOS digital circuits, there are basically two sources of power dissipation, dynamic power dissipation and static power dissipation. Dynamic power dissipation arises from the transitions of the CMOS gate. It includes switching power dissipation and short-circuit power dissipation. When the logic level of the logic gate switches between '0' (low voltage) and '1' (high voltage), the parasitic capacitances are charged and discharged, during which the energy is converted into heat dissipation when current flows through the transistor channel resistance. This type of power dissipation is called switching power dissipation, which consumes most of the power used by CMOS circuits. Short-circuit power dissipation happens during the short transition interval when the output of a gate is changing in response to the changes of inputs. During the transition, the n-subnetwork and p-subnetwork of a CMOS gate both conduct simultaneously and a current flow from voltage source directly to ground is incurred. Static power consumption in CMOS circuit is mainly due to leakage. When gates are not transitioning, and because they are not fully turned off, there is a static leaking current flowing through the transistors, causing static power dissipation. Leakage power dissipation also contributes an important portion of the total power dissipation. The total power dissipation of a circuit is the sum of the three power sources. We will introduce each of them in detail in the following subsections.

2.2.1 *Dynamic Power Dissipation*

The dynamic power dissipation due to the switching of CMOS circuit, P_{swi}, can be calculated by the following equation:

$$P_{\text{swi}} = \alpha C_{\text{load}} V_{\text{dd}} V_{\text{out}} f \tag{2.1}$$

where α is the switching activity, i.e., the fraction of clock cycles in which the circuit switches, C_{load} is the load capacitance switched as shown in the Fig. 2.2, V_{dd} is the supply voltage, V_{out} is the output voltage, and f is the clock frequency. Usually V_{out} is equal to V_{dd} in the CMOS digital circuit. From this we have:

$$P_{\text{swi}} = \alpha C_{\text{load}} V_{\text{dd}}^2 f \tag{2.2}$$

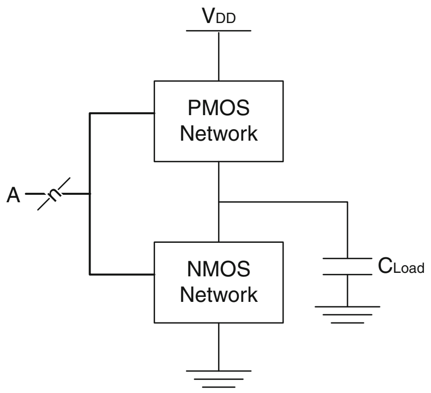


Fig. 2.2 CMOS gate with load capacitance

We can see from the equation that the larger the switched capacitance and the more frequently the circuit switches, the larger the dynamic power dissipation will be. Increasing V_{dd} will speed up the transistor transition, however, the switching power will increase quadratically with V_{dd}.

The energy consumed in the switching can be calculated based on the power consumption as:

$$\text{Energy} = \int p(t)\,\mathrm{d}t \tag{2.3}$$

where $p(t)$ is the power consumption of the circuit varying with time, which equals to the current from the source, I_{source}, times source voltage, i.e., $p(t) = I_{\text{source}}(t)V_{\text{dd}}$. Substitute $p(t)$ into Equation (2.3), and we obtain the expression for energy consumption as

$$\text{Energy} = \int I_{\text{source}}(t)V_{\text{dd}}\,\mathrm{d}t = \int C_{\text{load}}\frac{\mathrm{d_{out}}}{\mathrm{d}t}V_{\text{dd}}\,\mathrm{d}t = C_{\text{load}}V_{\text{dd}}\int d_{\text{out}} \tag{2.4}$$

When V_{out} is equal to V_{dd}, the energy drawn from the power supply is $C_{\text{load}}V_{\text{dd}}^2$.

Next, we will introduce how load capacitance is composed and how to obtain switching activities.

2.2.1.1 Components of Load Capacitance

First, we observed that in CMOS digital circuit, all gates drive other gates through on-chip interconnects. Figure 2.3a shows the capacitance in a MOSFET. We use Fig. 2.3b to illustrate the parasitic capacitance components that one gate usually drives. The overall load capacitance of the driver can be modeled as the parallel combination of the gate capacitances of the transistors that it drives C_{g}, the interconnect wire capacitance C_{int}, and its own drain-to-body capacitance C_{db}.

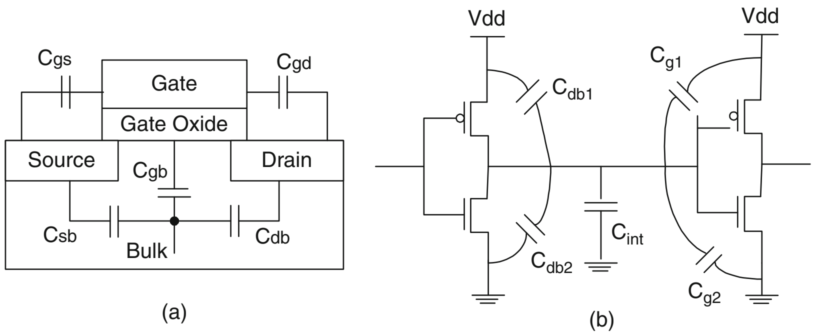


Fig. 2.3 **(a)** Capacitance in a MOSFET, **(b)** an inverter driving another showing parasitic load capacitance

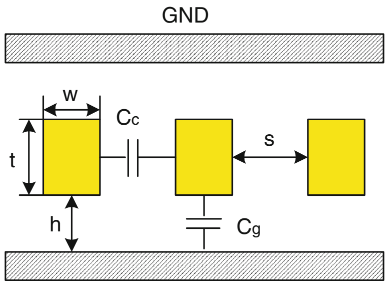


Fig. 2.4 Interconnect capacitance

Interconnect capacitance mainly comprises the area and fringe capacitance to the plane C_{g} and the coupling capacitance between neighboring wire C_{c}. Figure 2.4 shows the structure of the interconnect capacitance. The combined area and fringing-field capacitance C_{g} is given by [2]

$$C_{\mathrm{g}} = \varepsilon_{\mathrm{ox}} \left[\frac{w}{h} - \frac{t}{2h} + \frac{2\pi}{\ln\left(1 + \frac{2h}{t} 1 + \sqrt{1 + \frac{t}{h}}\right)} \right] \times l \tag{2.5}$$

$$w \geq \frac{t}{2} \tag{2.6}$$

where ε is the dielectric of the oxide insulation. l is the length of the metal wire. w, t, h, and s are the width and height of the wire, the distance of the wire to the plane, and the wire spacing as shown in the figure. The mutual-coupling capacitance between two wires can be calculated with different complexity [3].

We can see from the equation that the interconnect capacitance is proportional to the ratios between w, h, t, and s, which is determined by the design rule of the corresponding technology node. As the technology node continues scaling, there is a corresponding shrinking of the parameters. While the reduction of ε, w, and t

helps to reduce the total capacitance, C_g will increase with the decrease of h, and C_c increases with the decrease of s. As a result, the total interconnect capacitance first decreases with technology node scaling and then increases [4].

2.2.1.2 Switching Activity

Dynamic power consumption depends on the switching activity of the signals involved. In this context, the switching activity of the signals, α, can be defined as the average number of 0–1 transitions per clock cycle. Since there are equal probabilities to change from 0 to 1 and from 1 to 0, α is also equal to half of the total transitions of the node per cycle. If, in a time period with N clock cycles, the total number of switches for a signal is $n(N)$, then the switching activity of the signal can be calculated by $\alpha = \frac{n(N)}{2N}$. When gate outputs have glitches, they also contribute to the switching activities. Glitches are defined as the uncontrolled appearances of signal transitions. In some cases, it is found that the power consumption caused from glitches can account for up to 67% of the total dynamic power [5].

2.2.1.3 Clock Power Dissipation

Fully synchronous operation with clock signals has been the dominant design approach for digital systems. Recently, as technology nodes scale down to submicron regime, the clock frequencies of CMOS digital systems approach gigahertz range. Carrying large loads and switching at high frequency, clock power dissipation is a major source of the overall dynamic power dissipation in a digital system.

The dynamic power dissipated by switching the clock follows:

$$P_{\mathrm{clk}} = fV_{\mathrm{dd}}^2(C_{\mathrm{l}} + C_{\mathrm{d}}) \tag{2.7}$$

where C_{l} is the total load on the clock and C_{d} is the capacitance in the clock driver.

For example, for the H-tree-based global clock routing commonly used in digital circuits, as shown in Fig. 2.5, the source-to-sink delay is balanced to reduce the

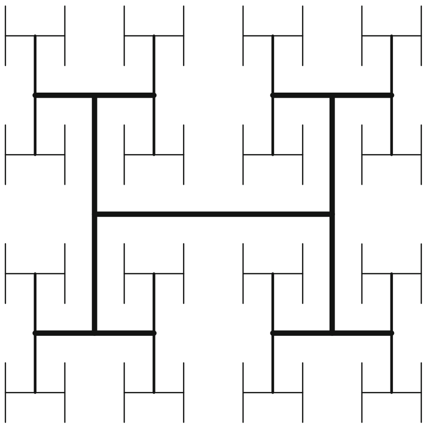


Fig. 2.5 H-tree clock distribution network

clock skew at the terminal. Given a total number of clock terminals N, the input capacitance at the terminal C_{in}, the unit length wire capacitance C_{wire}, the chip dimension D, and level of the H-tree h, its C_{l} can be estimated as [6]:

$$C_{\mathrm{l}} = NC_{\mathrm{in}} + 1.5(2^h - 1)DC_{\mathrm{wire}} + \alpha\sqrt{N4^h}C_{\mathrm{wire}} \tag{2.8}$$

It can be seen that the clock load capacitance, hence clock power dissipation, increases as the number of clocked terminals and the chip dimensions increase.

To ensure fast clock transitions, two common clock driving schemes, single driver or distributed buffer, are used to drive the large load capacitance on a clock. The single driver scheme uses a large buffer at the clock source. In this scheme, wire sizing is used to reduce the clock delay. Branches closer to the clock source are made wider. It also helps to reduce clock skew caused by asymmetric clock tree loads and wire width deviations [7, 8]. In the distributed buffer scheme, intermediate buffers are inserted in various parts of the clock tree. Relatively small buffers can be flexibly placed across the chip to save area and reduce clock delay. Figure 2.6 illustrates the two driving schemes. In any scheme, C_{l} and C_{d} need to be calculated for each buffer and summed to obtain the total capacitance for clock power dissipation.

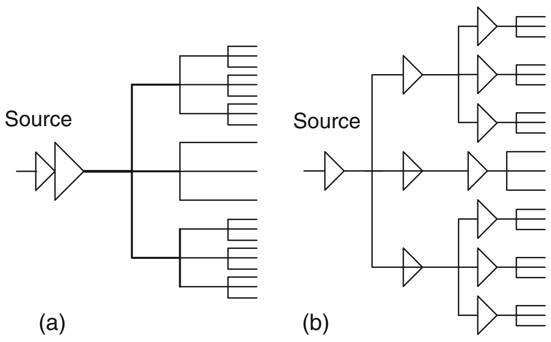


Fig. 2.6 Two clock tree driving schemes: **(a)** single driver scheme, **(b)** distributed buffers scheme

Traditionally, the load capacitance was largely contributed to by the capacitance of clock terminals. However, as technology advances into the deep submicron, device size is shrinking, while chip dimensions are increasing. This makes the interconnect capacitance dominant [6]. Reducing the interconnect capacitance may significantly reduce the overall power consumption. However, minimizing clock power consumption has to be considered together with meeting the constraints of clock skew and clock delay. Comparing the two clock schemes, the scheme of distributed buffers is preferred for low power design since it reduces both path length and load capacitance. Hence, it also reduces clock skew and minimizes the wire width. Thus, the total wire capacitance is kept at a minimum.

2.2.1.4 Short-Circuit Power Dissipation

The short-circuit power dissipation can be modeled using the following equation [9].

$$P_{\mathrm{sc}} = \frac{\beta}{12}(V_{\mathrm{dd}} - V_{\mathrm{tn}} - V_{\mathrm{tp}})^3 \frac{3\tau}{T} \tag{2.9}$$

where $V_{\mathrm{tn}}, V_{\mathrm{tp}}$ is the threshold voltage of the n and p transistor, respectively. β is the parameter determined by the rising and falling time of the output [9]. τ is the rise or fall time of the input signal, and T is the clock cycle of the input signal. However, since the gate may not switch during every clock cycle, the node activity factor must be added in. $\frac{1}{T}$ is revised to be $(\alpha_{10} + \alpha_{01})f$, where f is the input frequency [10]. On the other hand, later studies also show that P_{sc} closely varies with different load capacitances, and that the simple equation is mainly for slow input signals (large τ). Hence, various complicated models are derived to give a more accurate estimation of P_{sc} when needed [11, 12]. Since P_{sc} is normally less than 10% of the dynamic power, in most cases, it is neglected [10].

2.2.2 Leakage Power Dissipation

As a result of continued IC process scaling, which reduces transistor threshold voltage, channel length, and gate oxide thickness, the importance of leakage power consumption is increasing [12]. Presently, leakage accounts for 40% of the power consumption of modern 65 nm high-performance microprocessors, and will continue to increase as technology scales further [13]. Without leakage reduction techniques, this ratio will increase with further technology scaling. Indeed, the primary goal of high-k and metal gate research and development effort is to address the fast increasing MOS transistor gate leakage power dissipation [14]. The impact of circuit leakage effect on IC performance, power consumption, temperature, and reliability is fast growing. IC leakage power consumption, which was largely ignored in the past, has become a primary concern in low-power IC design. Leakage power must now be carefully considered and optimized during the entire IC design flow.

IC leakage current consists of various components, including subthreshold leakage, gate leakage, reverse-biased junction leakage, punch-through leakage, and gate-induced drain leakage [15], as shown in Fig. 2.7. Among these, subthreshold leakage and gate leakage are currently dominant, and are likely to remain dominant in the near future [1]. They will be the focus of our analysis.

Considering weak inversion drain-induced barrier lowering and body effect, the subthreshold leakage current of a MOS device can be modeled as follows [16]:

$$I_{\mathrm{subthreshold}} = A_{\mathrm{s}} \frac{W}{L} v_{\mathrm{T}}^2 \left(1 - \mathrm{e}^{\frac{-V_{\mathrm{DS}}}{v_{\mathrm{T}}}}\right) \mathrm{e}^{\frac{(V_{\mathrm{GS}} - V_{\mathrm{th}})}{n v_{\mathrm{T}}}} \tag{2.10}$$

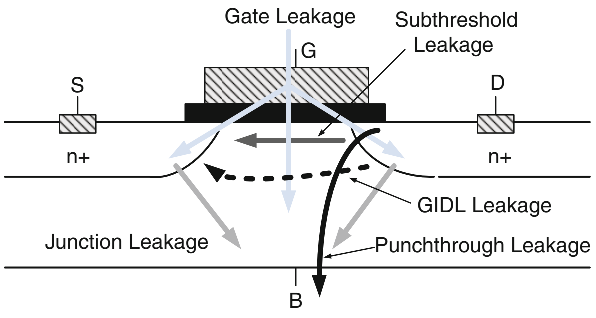


Fig. 2.7 Leakages current components in a MOS transistor

- where A_s is a technology-dependent constant,
- V_{th} is the threshold voltage,
- L and W are the device effective channel length and width,
- V_{GS} is the gate-to-source voltage,
- n is the subthreshold swing coefficient for the transistor,
- V_{DS} is the drain-to-source voltage, and
- v_T is the thermal voltage.

$V_{DS} \gg v_T$ and $v_T = \frac{kT}{q}$. Therefore, Equation (2.10) can be reduced to

$$I_{subthreshold} = A_s \frac{W}{L} \left(\frac{kT}{q} \right)^2 e^{\frac{q(V_{GS}-V_{th})}{nkT}} \tag{2.11}$$

Equation (2.11) shows that, IC subthreshold leakage current has exponential dependency on circuit threshold voltage. As described in Section 2.4.5, through technology scaling, stringent circuit performance requirements impose continuous reduction of IC threshold voltage, which results in significant increase of IC leakage current. As a result, IC leakage power consumption has become a first-order design issue, especially in the mobile application sector. Furthermore, IC subthreshold leakage current is a strong function of temperature. As temperature increases, circuit subthreshold leakage current increases superlinearly.

Leakage of a MOS device results from tunneling between the gate terminal and the other three terminals (source, drain, and body). Gate leakage can be modeled as follows [17]:

$$I_{gate} = WLA_J \left(\frac{T_{oxr}}{T_{ox}} \right)^{nt} \frac{V_g V_{aux}}{T_{ox}^2} e^{-BT_{ox}(a-b|V_{ox}|)(1+c|V_{ox}|)} \tag{2.12}$$

- where $A_J, B, a, b,$ and c are technology-dependent constants,
- nt is a fitting parameter with a default value of one,

- V_{ox} is the voltage across gate dielectric,
- T_{ox} is gate dielectric thickness,
- T_{oxr} is the reference oxide thickness,
- V_{aux} is an auxiliary function that approximates the density of tunneling carriers and available states, and
- V_g is the gate voltage.

Equation (2.12) shows that IC gate leakage current has strong dependency on device gate dielectric thickness. Through technology scaling, IC supply voltage reduces continuously in order to minimize circuit dynamic power consumption. However, to maintain good circuit performance, the thickness of transistor gate dielectric layer needs to reduce accordingly. Thus, IC gate leakage current increases exponentially. Modern transistor gate dielectric layer is only a few atoms thick. Gate leakage hence becomes a serious concern. To address this problem, intensive research and development efforts have been devoted searching for new gate dielectric material, i.e., high-k material, and the corresponding fabrication technology. Commercially solutions, such as metal-gate from Intel, have been widely deployed and demonstrated effective gate leakage reduction.

2.2.3 Limits of CMOS Circuits

Since the first transistor was fabricated in 1940s, the number of transistors per chip has continued to double every 18–24 months. This trend was observed by Gordon Moore of Intel, and is known as Moore's law. Accompanied by the growth in transistor density is the increase in reliability and decline of energy consumption per transistor transition. However, with continuous scaling, CMOS technology is approaching its scaling limits. Great attention is being paid to the limits of continued scaling [18–23]. Reference [20] defines a hierarchy of limits that have five levels: fundamental, material, device, circuit, and systems. Hu considers the reliability constraints on the scaling of MOS devices [19]. Currently, with CMOS scaling to below 32 nm, the primary obstacle arises from the static power dissipation, which is caused by leakage currents due to quantum tunneling and thermal excitations [22]. Next, we will discuss this power issue and the methods to overcome it.

2.2.3.1 Power-Constraint Scaling

Dynamic power can be adjusted to a limited extent by adjusting the load capacitance, the supply voltage, or the operational frequency. However, leakages, including subthreshold and gate-dielectric leakages, have unfortunately become the dominant barrier to further CMOS scaling, even for highly leakage-tolerant applications such as microprocessors.

As the channel length of a field effect transistor (FET) is reduced, the drain potential begins to strongly influence the channel potential, leading to drain-induced barrier lowering (DIBL). DIBL eventually allows electron flow between the source and the drain, even if the gate-to-source voltage is lower than the threshold voltage, leading to an inability to shut off the channel current with the gate. The channel current that flows under these conditions is called the subthreshold current. Subthreshold current contributes the most significant part of the static leakage consumption.

This short-channel effect (SCE) can be mitigated by reducing the gate oxide (to increase the control of the gate on the channel) and the use of thin depletion depth below the channel to the substrate, to shield the channel from the drain [24]. However, the reduction of gate oxide results in an increase of gate leakage current at the same time. At 90 nm CMOS, the power from gate leakage is comparable to the power used for switching of the circuit [24]. Thus, further reduction of gate oxide thickness would lead to unreasonable power increases. Alternatively, further decrease of the depletion region degrades gate control on the channel and slows down the turn on speed of the FET. For bulk CMOS, increased body doping concentration could be also employed to reduce DIBL; however, at some point it would also increase the subthreshold swing. Therefore, a higher threshold voltage is needed to keep the subthreshold current adequately low. Similarly, decreasing the body doping concentration could improve the subthreshold swing, but could degrade DIBL. Hence it is difficult to reduce both DIBL and subthreshold current for the bulk-silicon device design [24].

Besides gate tunneling leakage, other sources of tunneling leakage current are band-to-band tunneling between the body and drain of an FET and direct source-to-drain tunneling through the channel barrier [22].

In order to mitigate the impact of short channel effect to FET scaling, and reduce leakage power dissipation, new FET structure or even new technologies can be considered, such as carbon nanotube or graphene-based FETs.

2.2.3.2 Effects of Variations

Due to the limited resolution of photolithographic process, there is random process variation on the transistor dimensions from wafer to wafer and die to die around 10–20% [25], which increases with technology scaling. As discussed before, among the multiple leakage sources, subthreshold leakage and gate leakage play the most important role. According to the Equations (1.10), (1.11), (1.12), we can see that leakage is determined by the device dimensions (feature size, oxide thickness, junction depth, etc.), doping profiles, and temperature. Hence, as a result, statistical variation in each of the device parameters causes a large variation in each of the leakage components. For example, for subthreshold leakage, it linearly depends on the dimensions of the device and exponentially depends on the gate threshold voltage. Voltage on the other hand depends on oxide thickness, implant impurity, surface charge, etc. For gate leakage, it mainly depends on oxide thickness too exponentially. Hence the variation of gate oxide thickness will result in a large variation of leakage. To estimate the leakage current, we need to find its mean and standard variation using statistics. It can be done through either analysis or simulation. Reference [25] generalizes the statistical analysis procedure for estimating the mean *mu* and the standard deviation σ of a leakage component considering variation in a single parameter (say x) as the following:

- Express the current as function of the variable x as $g(x)$.
- Estimate mean of $g(x) : \mu[g(x)] = g(\mu_x + \frac{g^2(\mu_x)}{2}\sigma_x^2$.
- Estimate mean of $g(x)^2 : \mu[g(x)^2] = g^2(\mu_x) + \frac{(g^2)^2(\mu_x)}{2}\sigma_x^2$.
- Estimate standard deviation of $g(x) : \sigma[g(x)] = \sqrt{\mu[g(x)^2] - [\mu[g(x)]^2}$.

To derive the estimation of leakage component under variation of multiple parameters, according to the function of the variables, one can assume the relationship between variables to be independent or correlated, then apply the statistics equations for mean and standard deviation computation accordingly. The estimation can also be performed using Monte Carlo simulation method [26]. Using simulation method, the full chip level leakage estimation can be enabled with or without consideration of parameter correlation [27, 28].

Process variation degrades parametric yield by impacting both power consumption and performance of a design. This problem is enlarged by the fact that circuit timing is inversely proportional to power. For example, a reduction in channel length results in improved performance but also causes an exponential increase in leakage power. This inverse correlation makes it challenging to meet both power and frequency constraints. Many manufactured chips that meet timing end up exceeding power budget while other chips within the power limit fail to deliver the required performance [29]. Traditional parametric yield analysis of high-performance integrated circuits is mainly based on the frequency (or delay). For current design with power consumption as an important factor, integrated approachs to accurately estimate and optimize the yield when both a frequency and power limits are imposed on a design are proposed [29, 30].

2.3 Power Estimation

When designing VLSI circuits, the designers need to accurately estimate the silicon area, the expected performance, and power dissipation before fabricating the chip. Power estimation can be performed at different level resulting in tradeoff between estimation accuracy and efficiency. There are typically two types of power estimation: average power and peak power. Average power determines battery life while maximum or peak power is related to circuit cooling cost and reliability.

2.3.1 Dynamic Power Estimation

Since dynamic power consumption contributes a large portion of the total power consumption, dynamic power estimation is critical to the low-power design. Great amount of work has been conducted on dynamic power estimation on different design levels using simulation-based or analytic methods. It has been observed that when estimation is performed based on function blocks, the algorithm can produce result in a short time. However, when signal switch activities and internal load capacitances are required on individual lines for more accurate estimation, the results generating rate can be very slow. Next, we will introduce the commonly used estimation method for dynamic power estimation.

2.3.1.1 Simulation-Based Estimation

Circuit simulation-based techniques [31] simulate the circuit with a representative set of input vectors. This method is accurate and generous to different design. However, it suffers from the memory and execution constraint for large scale design. In general, it is difficult to generate a compact vector set, if it is not possible to go through all the input vectors, to calculate accurate activity factors at the circuit nodes.

To alleviate this problem, A Monte Carlo simulation approach for power estimation is proposed in [32]. This approach assumes randomly generated input patterns at the circuit inputs and simulate the power dissipation per time interval T. Based on the assumption that power dissipation over any interval has a normal distribution, given a error percentage and confidence level, the average power consumption is estimated. Note that the normality assumption may not hold in some cases, then the approach will converge to a wrong estimation.

2.3.1.2 Probabilistic Power Estimation

To estimate the dynamic power consumption, one has to calculate the switching activity of each internal node of the circuit. Assuming that the circuit inputs are independent to each other, probabilistic method can be used to estimate the switching activities of node n based on its signal probability prob(n). Then the switching activity of n is the probability of the signal switching from 0 to 1, which equals to $\alpha_n = \text{prob}(n)(1 - \text{prob}(n))$. If a network consists of simple gates and has no reconvergent fanout nodes, i.e., tree-like structure, then the exact signal probabilities can be computed during a single traversal of the network using the following equations [33]:

$$\begin{aligned}
&\text{not gate} : \text{prob(out)} = 1 - \text{prob(in)} \\
&\text{and gate} : \text{prob(out)} = \prod_{i \in \text{inputs}} \text{prob}(i) \\
&\text{or gate} : 1 - \prod_{i \in \text{inputs}} (1 - \text{prob}(i))
\end{aligned} \tag{2.13}$$

For network with reconvergent fanout, this simple algorithm yields approximate results for signal probabilities.

Another way for signal probability for general circuits is based on Shanon's expansion and BDD graph [34, 35]. According to Shanon's expansion,

$$f = x_i f_{x_i} + x_i' f_{x_i'} \tag{2.14}$$

where f_{x_i} means the function value when $x_i = 1$, i.e., $f(x_1, \cdots, x_{i-1}, 1, x_{i+1}, \cdots, x_n)$, and $f_{x_i'}$ correspondingly means the function value when substitute x with 0. Noted that $x_i, f_{x_i}, x_i', f_{x_i'}$ are independent with each other, the signal probability of f can be calculate as:

$$\begin{aligned} \alpha_f &= \text{prob}(x_i f_{x_i} + x_i' f_{x_i'}) \\ &= \text{prob}(x_i f_{x_i}) + \text{prob}(x_i' f_{x_i'}) \\ &= \text{prob}(x_i)\,\text{prob}(f_{x_i}) + \text{prob}(x_i')\,\text{prob}(f_{x_i'}) \end{aligned} \tag{2.15}$$

Boolean difference is another similar method for probabilistic switching activity estimation in combinational logic described in detail in [36]. Note that all the above estimation is based on zero-delay model. Estimation under a real delay model using symbolic simulation is presented in [37]. Above discussion mainly focuses on estimation for combinational logic. The estimation method for sequential circuit can greatly differ from the combinational ones. The average switching activity estimation for finite state machines (FSM) need to consider two impacts: (1) The probability of the circuit being in each of its possible states. (2) The probability of present state line inputs. The work in [38] presents the method to compute the exact state probabilities of the FSM using Chapman-Kolmogorov equations. For each state S_i in total K states, I_{ij} specify the input combination which transfers the FSM from state i to state j. Given static probabilities for the primary inputs to the machine, we can compute the conditional probability of going from S_i to S_j, $\text{prob}(S_j|S_i)$. For each state S_j, we can write the equation:

$$\text{prob}(S_j) = \sum_{S_i \in \text{instates}(S_j)} \text{prob}(S_i)\text{prob}(S_j|S_i) \tag{2.16}$$

where instate(S_i) is the set of fanin states of S_i in the state transfer graph. Given K states, we obtain K equations. Finally, we have a last equation:

$$\sum_j \text{prob}S_j = 1 \tag{2.17}$$

Different probabilities of the states can be obtained by solving the set of linear equations. We use an example in Fig. 2.8 to illustrate the procedure. Suppose all the FSM inputs are with signal probability 0.5. We can get the set of equations as:

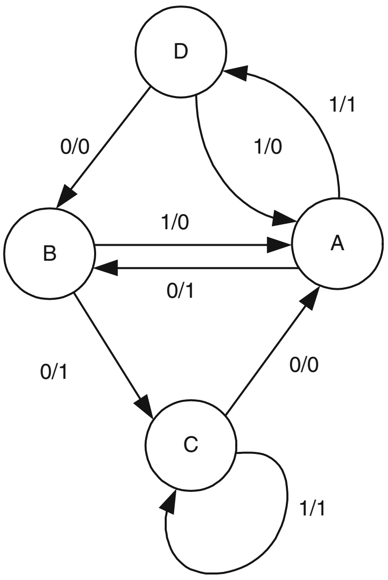


Fig. 2.8 State transfer graph for an example FSM

$$\begin{aligned}
\text{prob}(D) &= 0.5 \times \text{prob}(A) \\
\text{prob}(A) &= 0.5 \times \text{prob}(D) + 0.5 \times \text{prob}(B) + 0.5 \times \text{prob}(C) \\
\text{prob}(B) &= 0.5 \times \text{prob}(D) + 0.5 \times \text{prob}(A) \\
\text{prob}(A) &+ \text{prob}(B) + \text{prob}(C) + \text{prob}(D) = 1
\end{aligned} \tag{2.18}$$

By solving the set of equations, we can get $\text{prob}(D) = \frac{1}{6}, \text{prob}(A) = \frac{1}{3}$, $\text{prob}(B) = \frac{1}{7}$, and $\text{prob}(C) = \frac{1}{8}$.

2.3.1.3 Power Macromodeling

In high-level power estimation, the circuit is described as a set of blocks with known internal structure. The power dissipation of each block is estimated using a macromodel [39–41]. Here macromodel means some form of equation which can best match the simulated power dissipation numbers. The equation can only contain the variable of primary inputs, or the statistics can be used to improve the macromodel. For example, the statistics based macromodel can be:

$$\text{Power} = f(\text{Mean}_A, \text{Mean}_B, \text{SD}_A, \text{SDB}, \text{TC}_A, \text{TC}_B, \text{SC}_{AB}, gl_A, gl_B) \tag{2.19}$$

where A and B are the inputs to the block. Mean, SD, TC, SC, and gl indicate mean, standard deviation, time correlation, spatial correlation, and glitching factor, respectively. The macromodel can be built based on analytic derivation or simulation. A regression method for constructing macromodels are illustrated in Fig. 2.9 [41].

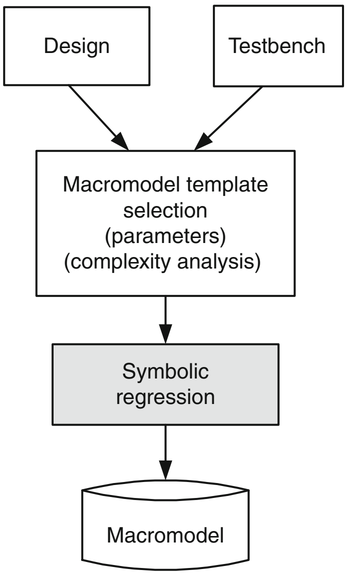

Design
Testbench
Macromodel template selection (parameters) (complexity analysis)
Symbolic regression
Macromodel

Fig. 2.9 Power macromodel generation using regression method

2.3.2 *Leakage Power Estimation*

Researchers have developed a variety of techniques to characterize IC leakage power consumption, ranging from architectural level to device level [15, 42–46]. We now survey leakage analysis work spanning these design levels.

Device-level leakage power estimation generally relies on models for individual transistor leakage mechanisms. Transistor leakage power consumption is a function of device physical properties and fabrication process. For bulk CMOS, the main control variables for leakage are device dimensions (feature size, oxide thickness, junction depth, etc.) and doping profiles in transistors [15]. Based on those physical characteristics, leakage models can be developed to predict the components of leakage, e.g., subthreshold leakage, gate leakage, and junction leakage [47]. Generally, technology constants provided by foundry can be used in such models. Transistor-level simulators incorporating these models can accurately predict leakage [48]; however they are computationally expensive due to iteratively solving complex leakage formulas.

In addition to its dependence on device parameters, IC leakage power consumption is affected by a number of circuit level parameters, e.g., the distribution of device types (NMOS and PMOS), geometries (channel width and length), and control voltages. Numerous circuit-level leakage estimation techniques have been proposed. Sirichotiyakul et al. presented an accurate and efficient average leakage calculation method for dual-threshold CMOS circuits that is based on graph reduction techniques and simplified nonlinear simulation [49]. Lee et al. proposed fast and accurate state-dependent leakage estimation heuristics using circuit block level look-up tables, targeting both subthreshold and gate leakage [50]. To conduct full-chip leakage estimation accurately, it is possible to model and sum the leakage currents of all gates [51, 52]. However, this is too computationally intensive for use in earlier design stages of very-large scale integration circuits.

For architectural leakage models [42], design parameters characterizing microarchitectural design styles and transistor sizing strategies can be extracted from typical logic and memory circuits. Do et al. proposed high-level dynamic and leakage power models to accurately estimate physically partitioned and power-gated SRAM arrays [53]. Given a set of inputs, Gopalakrishnan et al. used a bit-slice cell library to estimate the total leakage energy dissipated in a given VHDL structural datapath [54]. Kumar et al. presented a state-dependent analytical leakage power model for FPGAs [55]. The techniques described in this paragraph provide reasonable accuracy for early design stage leakage estimation, as long as temperature is fixed. However, they do not consider temperature variations.

IC leakage power consumption is a strong function of temperature, e.g., subthreshold leakage increases super-linearly with chip temperature. In modern microprocessors, power density has reached the level in a nuclear reactor core, causing high chip temperatures and hence high leakage power consumptions. Due to time-varying workload and operating states with different power consumption levels (up to 25 power states in the Core$^{\text{TM}}$ Duo processor [56]) and uneven on-chip power density distribution, large on-chip temperature variations and gradient are common in high-performance ICs. For example, ICs may have larger than 40°C temperature difference [57], causing high on-chip leakage variation. In summary, increasing chip temperature and on-chip temperature variation significantly affect IC leakage power consumption [58]. Therefore, accurate leakage power analysis requires temperature to be considered.

Some researchers have developed temperature-dependent architectural leakage power models. Zhang et al. developed HotLeakage, a temperature-dependent cache leakage power model [59]. Su et al. proposed a full-chip leakage modeling technique that characterizes the impact of temperature and supply voltage fluctuations [60]. Liao et al. presented a temperature-dependent microarchitectural power model [61].

Figure 2.10 shows a typical temperature-aware power estimation flow. Power consumption, including dynamic power and leakage power, is initially estimated at a reference temperature. Given an IC design, initial dynamic power for each circuit macro block is determined by estimated macro block workloads, switching factors, and supply voltage using commercial tools such as Synopsys Power Compiler and PrimePower in its Galaxy Design Platform [62]. Initial leakage power can be obtained by HSPICE simulation or other efficient high-level methods [49, 51, 52]. The estimated power profile is then provided to a chip-package thermal analysis tool to estimate circuit thermal profile. This thermal profile is, in turn, used to update circuit macro block leakage power. This iterative process continues until leakage power consumption converges.

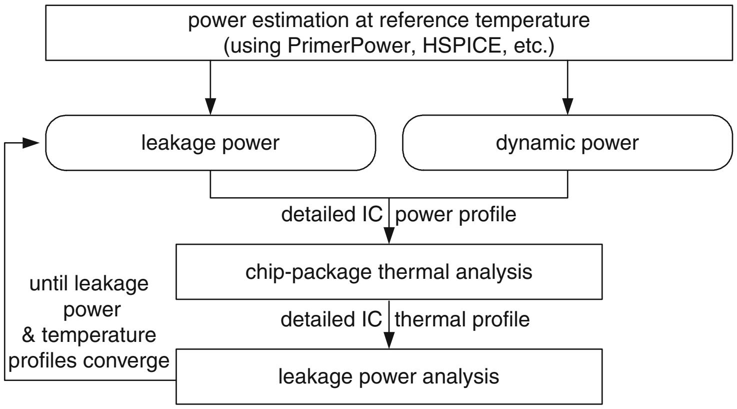

power estimation at reference temperature
(using PrimerPower, HSPICE, etc.)
leakage power
dynamic power
detailed IC power profile
chip-package thermal analysis
detailed IC thermal profile
leakage power analysis
until leakage power & temperature profiles converge

Fig. 2.10 Temperature-aware leakage power analysis flow

2.3.2.1 Thermal Analysis

IC thermal analysis is the simulation of heat transfer through heterogeneous material among heat producers (e.g., transistors) and heat consumers (e.g., heat sinks attached to IC packages). Modeling thermal conduction is analogous to modeling electrical conduction, with thermal conductivity corresponding to electrical conductivity, power dissipation corresponding to electrical current, heat capacity corresponding to electrical capacitance, and temperature corresponding to voltage.

The equation governing heat diffusion via thermal conduction in an IC follows.

$$\rho c \frac{\partial T(\mathbf{r},t)}{\partial t} = \nabla (k(\mathbf{r}) \nabla T(\mathbf{r},t)) + p(\mathbf{r},t) \tag{2.20}$$

subject to the boundary condition

$$k(\mathbf{r},t)\frac{\partial T(\mathbf{r},t)}{\partial n_i} + h_i T(\mathbf{r},t) = f_i(\mathbf{r},t) \tag{2.21}$$

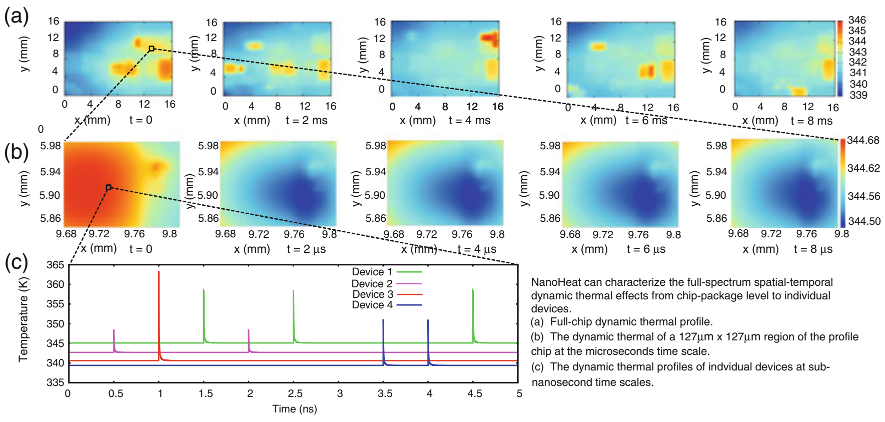
In Equation (2.20), ρ is the material density, c is the mass heat capacity, $T(\mathbf{r},t)$ and $k(\mathbf{r})$ are the temperature and thermal conductivity of the material at position $\mathbf{r}$ and time t, and $p(\mathbf{r},t)$ is the power density of the heat source. In Equation (2.21), n_i is the outward direction normal to the boundary surface i, h_i is the heat transfer coefficient, and f_i is an arbitrary function at the surface i. Note that, in reality, the thermal conductivity, k, also depends on temperature. Recently proposed thermal analysis solutions begin to support arbitrary heterogeneous three-dimensional thermal conduction models. For example, a model may be composed of a heat sink in a forced-air ambient environment, heat spreader, bulk silicon, active layer, and packaging material or any other geometry and combination of materials.

In order to do numerical thermal analysis, a seven point finite difference discretization method can be applied to the left and right side of Equation (2.20), i.e., the IC thermal behavior may be modeled by decomposing it into numerous rectangular parallelepipeds, which may be of non-uniform sizes and shapes. Adjacent elements interact via heat diffusion. Each element has a power dissipation, temperature, thermal capacitance, and a thermal resistance to adjacent elements. For an IC chip-package design with N discretized elements, the thermal analysis problem can then be described as follows.

$$\mathbf{C}T(t)' + \mathbf{A}T(t) = Pu(t) \tag{2.22}$$

where the thermal capacitance matrix, $\mathbf{C}$, is an $[N \times N]$ diagonal matrix; the thermal conductivity matrix, $\mathbf{A}$, is an $[N \times N]$ sparse matrix; $T(t)$ and $P(t)$ are $[N \times 1]$ temperature and power vectors; and $u(t)$ is the time step function. For steady-state analysis, the left term in Equation (2.22) expressing temperature variation as function of time, t, is dropped. For either the dynamic or steady-state version of the problem, although direct solutions are theoretically possible, the computational expense is too high for use on high-resolution thermal models.

Both steady-state and dynamic analysis methods have been developed in the past for full-chip IC thermal analysis [60, 63, 64, 65–69].These solutions, however, are unable to support nanometer-scale device-level spatial and temporal modeling granularities. Furthermore, they rely on the Fourier thermal physics model. The Fourier model with fixed material thermal conductivities fails at length scales comparable to the phonon mean free path (the average distance phonons travel before suffering scattering events), and at time scales comparable to that on which phonon scattering events occur [70]. Current device sizes and switching speeds have already reached those limits. This yields the Fourier equation inadequate for modeling device-level thermal effects. Recently, Allec et al. [71] proposed a multi-scale IC thermal analysis solution that can characterize the device-level thermal effect, producing IC thermal profiles with transistor-level spatial resolution. Figure 2.11 shows the runtime thermal profile of an IC design. It shows that, from chip–package level to



NanoHeat can characterize the full-spectrum spatial-temporal dynamic thermal effects from chip-package level to individual devices.
(a) Full-chip dynamic thermal profile.
(b) The dynamic thermal of a 127µm x 127µm region of the profile chip at the microseconds time scale.
(c) The dynamic thermal profiles of indvidual devices at sub-nanosecond time scales.

Fig. 2.11 Thermal characterization of nanometer-scale ICs

individual devices, the IC thermal characteristic is heterogeneous and dynamic, which has direct impact on IC performance, power consumption, short-time scale, and lifetime reliability.

2.4 Power Optimization and Management

In order to meet the power requirement, lengthen the battery life, and increase the chip reliability, power optimization has become a crucial design aspect. Power optimization can also be performed at different design phases, targeting dynamic power reduction or leakage power reduction. The higher the level power optimization is performed, the more power reduction that can be obtained. In the following subsections, we will discuss the optimization procedures for low power dissipation at algorithm, architecture, gate, and circuit levels.

2.4.1 High-Level Synthesis for Low Power

As we have discussed in Section 2.2.1, the dynamic power dissipation is proportional to the value of switching activity, load capacitance, and source voltage. Hence, in low power design, we will reduce the power dissipation of circuits by reducing each of the factors first in the high level synthesis. Due to the large freedom for synthesis from high-level specification to low-level implementation, there is great potential of producing a large improvement in power dissipation during the synthesis procedure. At the behavioral level, since operations have not been assigned, the execution time and hardware allocation have not been performed, a systematic design space exploration can be performed to search for a design point satisfying the given power dissipation, delay, and area constraints. Traditionally, behavioral synthesis has targeted optimization of hardware resource usage and average clock cycles for execution of a set of tasks. With the increase in concern for power consumption, a great amount of studies have been carried out to examine the efficacy of behavioral level techniques to reduce power dissipation. The essential methodology is to reduce the number of power-consuming operations, restructure the microarchitecture to reduce switching activities, and balance the microarchitecture to reduce glitches. Reference [72] proposes an iterative algorithm for performing scheduling, clock selection, module selection, and resource allocation and assignment simultaneously with an aim of reducing the power consumption in the synthesized data path. Next, we introduce some of these techniques in detail.

2.4.1.1 Transformation

Some parts of the CDFG can be restructured to reduce the number of modules used, and hence, reduce the power consumption needed to realize the same functionality. For example, in Fig. 2.12a, two multipliers will switch in one step. However, after

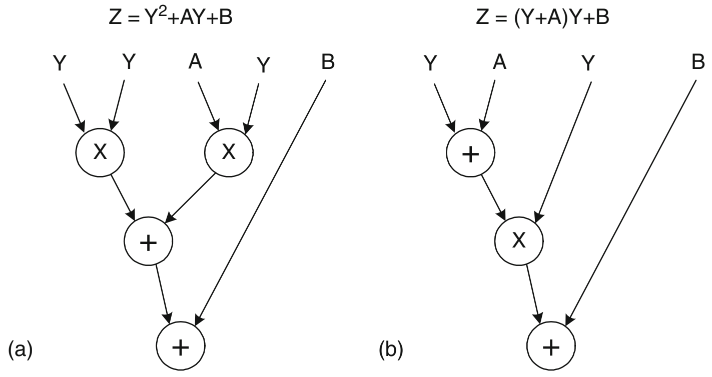


Fig. 2.12 Restructure of the circuit to reduce power consumption

a slight structural transformation, only one multiplier switches in one step as in Fig. 2.12b.

2.4.1.2 Scheduling

The behavioral level synthesis is based on the behavioral description of the circuit in the form of a control-dataflow graph (CDFG). A CDFG is a directed graph whose vertices consist of arithmetic, logical and comparison operations, delay operators, and special branch, merge, loop entry, and loop exit vertices that represent control flow constructs [72]. The CDFG contains data (control) flow edges that represent data (control) dependencies between operations. An example CDFG shown in Fig. 2.13a represents the computation of $z = |a - b|$. The process of scheduling assigns each operation in the CDFG to one or more cycles or control steps. Figure 2.13b, c illustrate two possible schedules of the task graph using minimum resources. The horizontal dotted lines labeled with numbers indicate the clock edges, i.e., the boundaries between control steps. The task takes three clock cycles to execute. In Fig. 2.13b, since two subtraction operations are preformed in different clock cycles, one subtractor is sufficient to implement the functions, i.e., executing subtraction 1(2) in clock 1(2). For the scheduling in Fig. 2.13c, although subtraction 1 and 2 are in one step, they are mutually exclusive. They will not execute at the same time; hence, they can be realized by one subtractor. Therefore, both schedules use one comparator, one MUX and one subtractor. However, the subtractor will be active in both clocks in the first scheduling, while in the second scheduling, the subtractor is only active during one clock and hence will consume less power.

2.4.1.3 Module Selection

Module selection refers to the process of selecting, for each operation in the CDFG, the type of functional unit that will perform the operation. In order to fully explore the design space, it is necessary to have a diverse library of functional units which

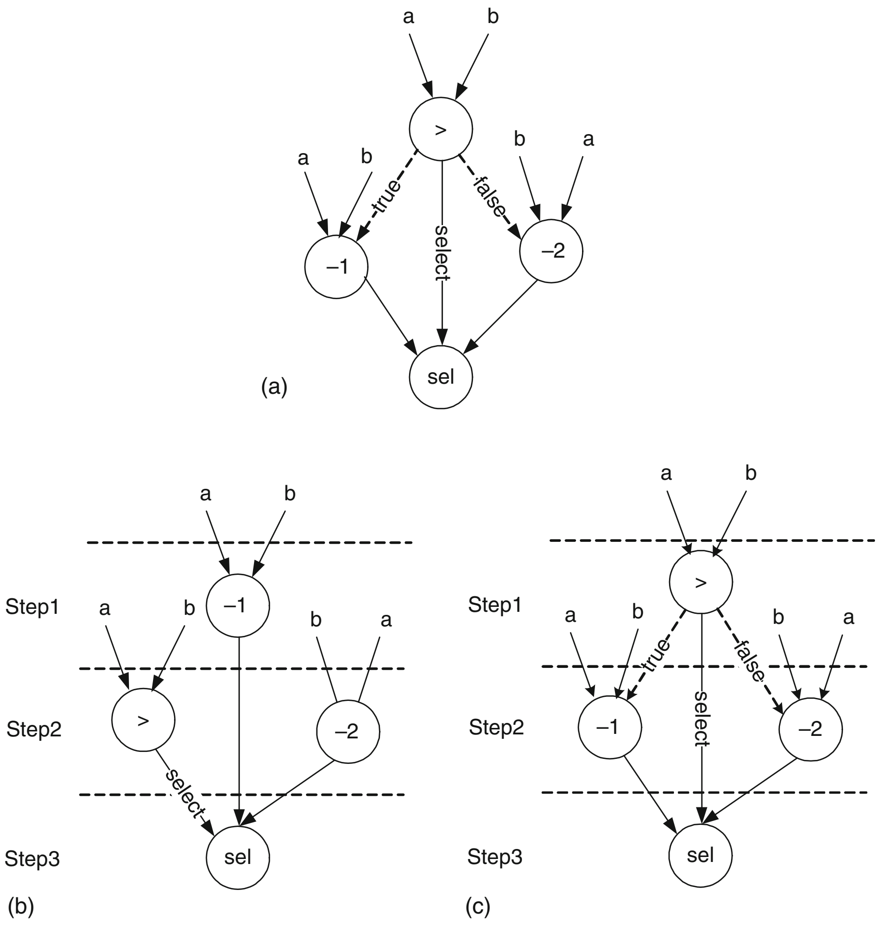


Fig. 2.13 Different schedules for the example circuit

perform the same function; however, do so with different logic implementation, and hence, different area, delay, and power consumption. For example, there can be many types of adders including ripple-carry-adder, carry-lookahead-adder, carry-select-adder, etc. A faster module is typically more expensive in terms of area and switched capacitance, i.e., power consumption. It is possible to perform tradeoffs in area, delay, and power using module selection. Suppose there are two types of multipliers, a WAL-MULT with 40 ns delay and an ARR-MULT with 60 ns delay. The power consumption of the ARR-MULT is much less than the WAL-MULT. When performing module selection as illustrated in Fig. 2.14, by replacing the WAL-M module with an ARR-M, the multiplication will take multiple clock cycles; however, the delay of the circuit will not be affected and power consumption is saved.

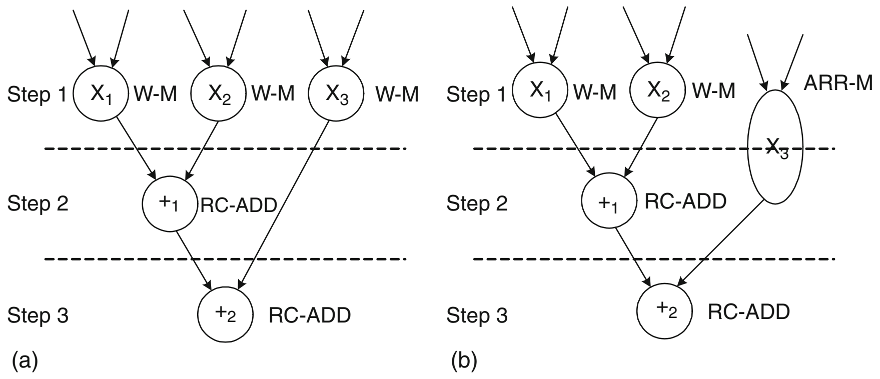


Fig. 2.14 Module selection for low power

2.4.1.4 Resource Sharing

Resource sharing refers to the use of the same hardware resource (functional unit or register) to perform different operations or store different variables. Resource sharing is performed in the step of hardware allocation and assignment, such as register binding, in the behavioral synthesis flow. These processes decide how many resources are needed and which type of resource to use to implement an operation or variable storage. Scheduling and clock selection, discussed later, affect how resource sharing is performed. In turn, resource sharing significantly affects both the physical capacitance and switching activity in the data path. Heavy resource sharing tends to reduce the number of resources used and the physical capacitance, but increases the average switching activity in the data path. Sparsely shared architectures have lower average switching activity, but higher physical capacitance and area usage. A detailed analysis of the effect of resource sharing on switched capacitance can be found in [72].

2.4.1.5 Clock Selection

Clock selection refers to the process of choosing a suitable clock period for the controller/data path circuit. Given the clock period, the execution time of the CDFG, which is equal to the input sample period, will be divided into several clock cycles. The choice of the clock period is known to have a significant effect on both area and performance [73]. However, it also has a great impact on power consumption as pointed out by [74]. A longer clock period is helpful for reducing the switching power consumption in the clock network, reducing the number of registers needed, and using modules with longer delay but less power consumption. However, on the other hand, a long clock period will contain more modules in its cycle, which tends to create more glitches between modules and inhibit the amount of resource sharing. Larger time slacks also tend to appear in long clock periods.

2.4.2 Architectural Level Power Optimization

After the circuit specification is synthesized into architecture-level functional units, several optimization methods aiming to reduce switching activity, source voltage or load capacitance can be performed based on the microarchitecture, and its performance, as discussed below.

2.4.2.1 Architecture-Driven Voltage Scaling

As seen from the equation for dynamic power dissipation, a large reduction in power dissipation can be obtained if the supply voltage is scaled down, as V_{dd} appears as a squared term in the expression. However, one direct side effect is the increase in circuit delay.

$$\mathrm{Delay} = \frac{kV_{\mathrm{dd}}}{(V_{\mathrm{dd}} - V_{\mathrm{t}})^{\alpha}} \tag{2.23}$$

where k and α are some parameters. One scenario to use voltage scaling is when there is time slack in the pipeline after behavioral synthesis. Another simple way to maintain throughput while reducing the supply voltage is to leverage parallelism or use a more deeply pipelined architecture. For example, the original circuit is as in Fig. 2.15a. If the combinational logic is further divided into N pipelines as in Fig. 2.15b, the voltage can be scaled down to close to N (equal to N, if V_{t} is zero), while throughput is not affected. However, note that by inserting more pipeline registers, total switching capacitance also increases. Finally, a less than N times power reduction can be achieved. With the price of area increasing, parallel architectures can reduce power dissipation even further. As illustrated in Fig. 2.15c, a parallel architecture duplicates the input registers and functional units to parallel process N samples, however, only one output register stores the corresponding result selected from the N units in one clock cycle. Since one unit only processes one sample during N clock cycles, the circuit delay for the unit can be N times the original circuit delay. By calculation, it is obtained that the source voltage can be scaled to $1/N$ of the original voltage. Then the power consumption of the parallel architecture becomes $\frac{1}{N^2}$ of the previous power consumption, as shown below.

$$\begin{aligned} P_{\mathrm{para}} &= NC_{\mathrm{load}}\frac{V_{\mathrm{dd}}^2}{N^2}\frac{f_{\mathrm{sample}}}{N} \\ &= \frac{1}{N^2}C_{\mathrm{load}}V_{\mathrm{dd}}^2 f_{\mathrm{sample}} \end{aligned} \tag{2.24}$$

2.4.2.2 Power Management with Clock Gating

On any given clock cycle, a part of the architecture may be idle. If so, it is possible to save substantial power by disabling clocks to the areas of the architecture that are

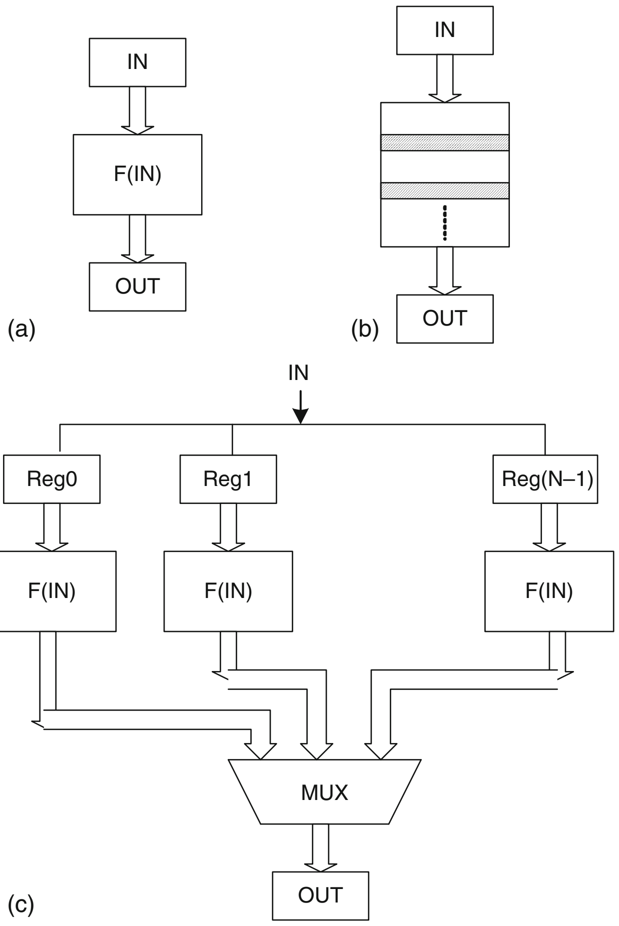


Fig. 2.15 Architectures enabling voltage scaling

idle. This saves power dissipation both in the clock tree and in the registers for when they do not need to change. Clock gating may be applied at any level of the design hierarchy, from a small logic block to the entire chip. Figure 2.16 illustrates how a register is gated. When the gating condition is true, the output of the OR gate will be 1, the D flip-flop value will not change, and therefore the following logic will not

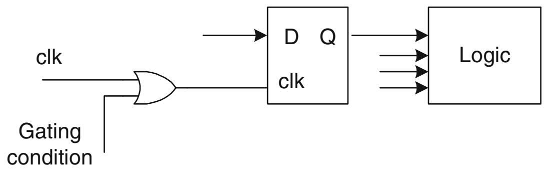


Fig. 2.16 Illustration of clock gating

switch. However, note that the more fine-grained the design is clock gated, the more complicated the control logic and hence, the more area overhead there will be.

2.4.3 Logic Level Optimization

Once various system level, architectural and technological choices are made, logic synthesis is performed to generate the netlist of gate specifications. During logic synthesis, different techniques are applied to transform the original RTL description to optimize for a specified objective function such as area, delay, power or testability. After logic synthesis, the power consumption of a circuit will be determined by the switched capacitance of the logic. In this section, a number of techniques for power estimation and minimization during logic synthesis for combinational and sequential circuits will be introduced. The strategy for low power logic synthesis is to obtain low switching activity factors and reduce the capacitive load at the nodes.

2.4.3.1 Multi-Level Network Optimization

Multi-level optimization of a Boolean network involves creating new intermediate signals and/or removing some of the existing ones to reduce area or improve performance. By adjusting the cost function targeting power consumption, multi-level network optimization techniques for low power design are also proposed [75, 76].

Don't care optimization is one way to reduce the switching activities. We use an example in Fig. 2.17 to illustrate the method. The input and output relation can be analyzed as follows: when $a = 1, f = c$; when $a = 0, f = 0$. Hence, c can only be observed at *f* when $a = 1$. However, when $a = 1, c = 1$. It can be seen that the circuit is equivalent to $f \equiv a$. b can be regarded as a don't care signal and assigned to be 1. The key point for don't care optimization is to identify don't care signals and minimize the switching activities of the circuits using them. Note that, when the probability for a signal to be 1, prob(1), is 0.5, its probable switching activity from 0 to 1 is the largest (prob(1) × (1 − prob(1))). Considering how changes in the global function of an internal node affects the switching activity of nodes in its fanout, [76] presents a greedy network optimization procedure. The optimization procedure works from the circuit outputs to the inputs, simplifying the fanouts of nodes. Once a node n is simplified, the procedure propagates those don't care conditions which help reduce the switching activity of its connected nodes. If the signal probability

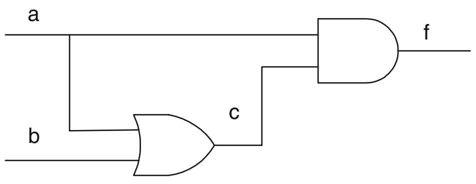


Fig. 2.17 Example for optimization with don't care conditions

of the node is greater than (less than or equal to) 0.5, don't care conditions are used to increase (decrease) the signal probability. Power consumption in a combinational logic circuit has been reduced by around 10% as a result of this optimization [10].

2.4.3.2 Logic Factorization

Sharing of common sub-expressions across the design reduces the number of literals and area, and may also improve the circuit performance. Based on the input of a set of Boolean functions, a procedure called kernel finds the common divisors to two or more functions. The best few common divisors are factored out, and the affected functions are simplified. This process is repeated until no common divisors can be found. The global area optimization process of SIS [77] has been proven to be very well suited for extensions to power optimization. The kernel extraction procedure is modified in [78] to generate multi-level circuits with minimized power.

An example of extraction is shown in Fig. 2.18. It depicts two ways of decomposition of the same function $f = ab + ac + bc$. Power consumption of decomposition A is equal to $P_{\text{decomp}(A)} = 2P_a + 2P_b + P_c + P_g + P_f$, and power consumption of decomposition B is equal to $P_{\text{decomp}(B)} = P_a + 2P_b + 2P_c + P_h + P_f$. Then $P_{\text{decomp}(A)} - P_{\text{decomp}(B)} = P_a + P_g - P_b - P_h$. According to the signal probabilities of the inputs, the decomposition with lower power consumption will be selected. However, it needs to noted that the computation for finding the optimal divisor for power minimization is expensive. Therefore, it is desirable to calculate only a subset of the candidate divisors.

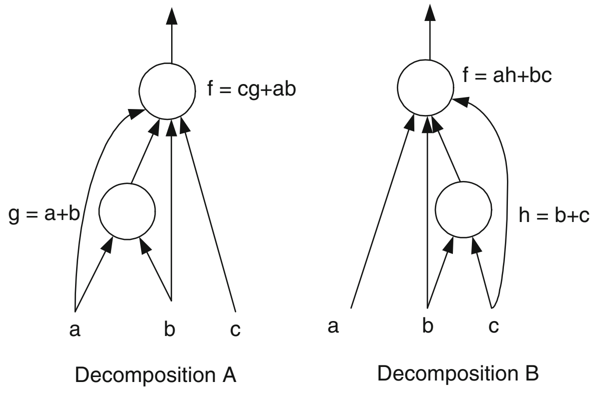


Fig. 2.18 Logic decomposition for low power

2.4.3.3 Path Balancing to Reduce Glitches

Balancing path delays reduces glitches in a circuit, which in turn reduces circuit dynamic power dissipation. This can be achieved through balanced logic structuring or delay insertion. Figure 2.19 shows an example of how logic structuring can balance an input path to reduce glitches. In (a), assuming that all the primary inputs

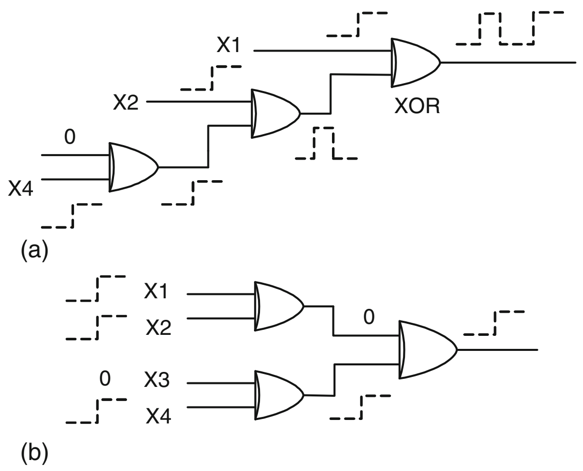


Fig. 2.19 Logic restructuring of a circuit to reduce glitches

arrive at the same time, different arrival times arise for the inputs to the XOR gates, due to the long path. This results in glitches in the internal lines and circuit output. By restructuring the logic as in (b), the input paths to the gates are balanced and such glitches are removed.

Another way to balance paths is to insert buffers to balance the path delay. Figure 2.20 illustrates one example for buffer insertion. The key issue in buffer insertion is to use the minimum number of buffers to achieve the maximum reduction in glitches.

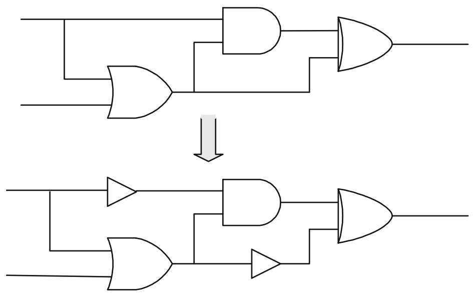


Fig. 2.20 Balance the path by inserting buffers

2.4.3.4 Technology Mapping

Technology mapping is the procedure of binding a set of logic equations (or Boolean network) to gates in the target cell library such that a given objective is optimized [77]. The objective can target area, performance, or power optimization. It can be solved by using the tree covering problem, as discussed in [79]. The circuit is decomposed into trees, and the tree covering problem is applied on all the

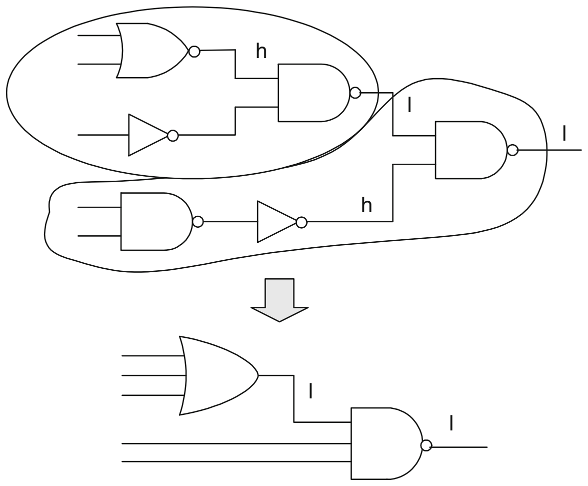


Fig. 2.21 Technology mapping for low power

trees separately. The tree covering problem can be solved efficiently with dynamic programming. The problem of minimizing average power consumption during technology mapping is addressed in [80–83]. The general principle is to hide nodes with high switching activity inside the gates that drive smaller load capacitances, as shown in Fig. 2.21. Lin and de Man [82] have considered power dissipation in the cost function for technology mapping. The formula for power estimation is given by

$$\text{Power} = \sum_{i=1}^{i=n} \alpha_i C_i f V_{\text{dd}}^2 \tag{2.25}$$

where f is the clock frequency and C_i, α_i are the load capacitance, and 0–1 switching activity is associated with the node i. The key point of dynamic programming approach is to break the optimization problem down into similar substeps that estimate the partial cost of intermediate solutions. Hence, the cost function for power dissipation during tree mapping can be formulated as

$$\text{power}(g,n) = \text{power}(g) + \sum_{n_i \in \text{inputs}(g)} \text{MinPower}(n_i) \tag{2.26}$$

where power(g) is the power contribution of choosing gate g to implement node n. The second term is the sum of minimum power costs for the corresponding subtrees rooted at the input pins of g. To minimize power consumption under a delay constraint, [80] proposes an approach composed of two steps. In the first step, the curve of power consumption versus arrival time at all nodes is computed and associated with the nodes. In the second step, a reverse pass from the root of the tree is performed to select the mapping with minimum cost that satisfies the delay constraint.

Note that under a real delay model, the dynamic programming-based tree mapping algorithm does not guarantee to find an optimal solution. The extension to a real delay is considered in [84].

2.4.3.5 State Assignment

We have discussed several methods for power optimization of combinational logic. Next, we will target power minimization in sequential logic. A finite state machine (FSM) is shown in Fig. 2.22. The first method addresses how to assign the states of an FSM to minimize the total number of state transitions (from current state S_1 to next state S_2), given the signal probability of primary inputs. The state assignment algorithm tries to minimize the activity for number of transitions at the present state inputs to the state machine. The basic assumption is that the higher the state transition activities at the input to a combinational circuit, the higher the rate at which the internal nodes of the circuit will switch, and hence more power is consumed. The approach to minimize the state transitions is to give uni-distance codes to states with high transition frequencies to one another [78]. The average number of switches at the present state inputs to a state machine can be modeled as:

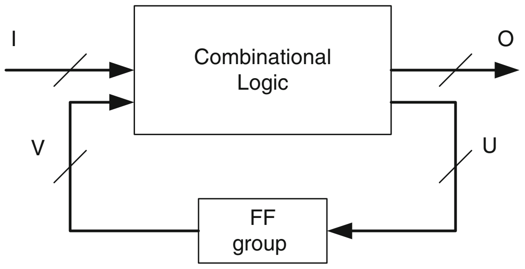


Fig. 2.22 State machine representation

$$N_{\text{swi}} = \sum_{\text{overalledges}} \text{prob}_{ij} H(S_i, S_j) \qquad (2.27)$$

where prob_{ij} is the probability of transaction from state i to state j. $H(S_i, S_j)$ is the Hamming distance between state i and j. For example, the probability of state transaction for the state assignment in Fig. 2.23a is $(0.5+0.8)\times 2+0.4+0.6+0.5+0.2 = 4.3$. By changing the state assignment and reducing the Hamming distance between the states with high switching probability, the total state transactions are reduced to $(0.4+0.5)\times 2+0.5+0.8+0.6+0.2 = 3.9$ as shown in Fig. 2.23b. The optimization can be achieved by iterative solution, such as simulated annealing.

However, the above objective function ignores the power consumption in the combinational logic that implements the next state. A more effective approach [85] considers the complexity of the combinational logic resulting from the state assignment and modifies the objective functions to achieve lower power dissipation. Genetic algorithm-based approaches [86–88] are also proposed for FSM synthesis

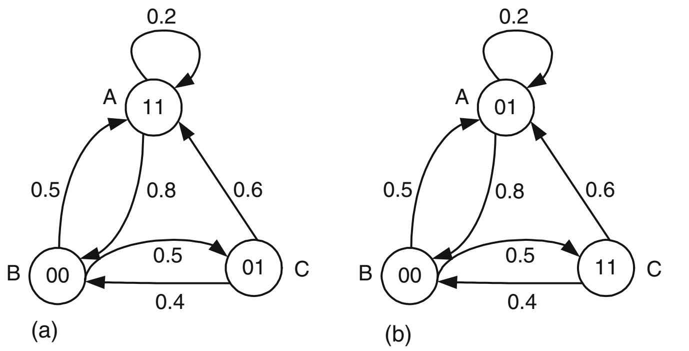


Fig. 2.23 Two different state assignments of the FSM

and state assignment, targeting low power dissipation. The approach considers both register switching and gate switching, and significant (over 10%) power savings can be obtained as compared to conventional encoding schemes, such as NOVA [89] and JEDI [90].

2.4.3.6 Precomputation

The basic idea for precomputation is to selectively compute some logic outputs and use them to reduce the internal switching activity in the succeeding clock cycle. A precomputation architecture is shown in Fig 2.24. The inputs to combinational logic have been partitioned into two sets, stored in registers R_1 and R_2. The output of the logic feeds register R_3. Boolean functions g_1, g_2 are the predictor functions. The precomputation conditions are that

$$
\begin{aligned}
g_1 = 1 &\Rightarrow f = 1 \\
g_2 = 1 &\Rightarrow f = 1
\end{aligned}
\tag{2.28}
$$

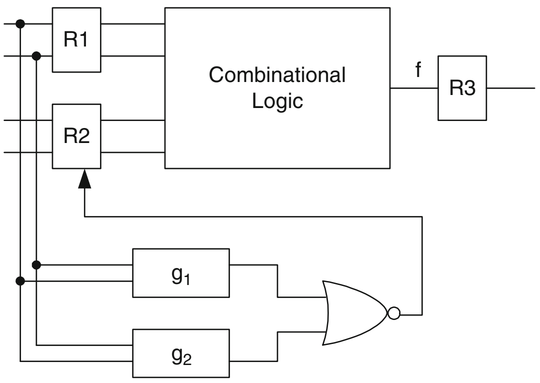


Fig. 2.24 A precomputation architecture

It means that if either g_1 or g_2 is evaluated to be 1, we can set the load enable signal of register R_2 to 0. This implies that the set of inputs will not be loaded to R_2. Since R_1 is still updated, function *f* will get the correct value. Since register R_2 and part of the combinational logic block are not switched, power reduction is obtained.

One example of usage of precomputation is the *n*-bit comparator that compares two *n*-bit numbers *A* and *B*. When the MSB $A_n > B_n \Rightarrow A > B$ and $A_n < B_n \Rightarrow A < B$. The result can be correctly predicted regardless of the left input bits.

Another way to obtain the predictor functions is to apply Shannon's decomposition. According to Shannon's decomposition theorem,

$$f = x_i f_{x_i} + x_i' f_{x_i'} \tag{2.29}$$

Define $U_{x_i} f = x_i f_{x_i} \cdot x_i' f_{x_i'}$ to be the universal quantification of *f* and x_i, and we can get the following equivalence.

$$U_{x_i} f = 1 \Rightarrow f_{x_i} = f_{x_i'} = 1 \Rightarrow f = x_i \cdot 1 + x_i' \cdot 1 = 1 \tag{2.30}$$

Similarly, it can be derived as

$$U_{x_i} f' \Rightarrow f'_{x_i} = f'_{x_i'} = 1 \Rightarrow f_{x_i} = f_{x_i'} = 0 \Rightarrow f = 0 \tag{2.31}$$

Hence, for function *f* with *m* input variables, there can be *m* predictor functions, i.e., $g_i = U_{x_i} f$. Power reductions of up to 40% have been reported in some examples. However, if the computation of a predictor is too complex, long delays may be invoked.

2.4.3.7 Retiming

Retiming is the process of re-positioning the flip-flops in a pipelined circuit in order to either minimize the number of flip-flops or minimize the delay of the longest pipeline stage. It is noted that an unbalanced path will cause glitches in combinational logic; however, the output of a flip-flop only makes one transition when the clock is asserted, as illustrated in Fig. 2.25. Based on this observation, retiming techniques are proposed that target low power dissipation [91, 92]. We can see in Fig. 2.25 that glitches are limited at the local capacitance C_K. However, since C_L is much larger than C_K, power dissipation due to glitches is greatly saved. The idea is to identify circuit nodes with high glitches and high load capacitance as proper candidates for adding flip-flops. The flip-flops also can be shifted around to help reduce the glitches at nodes with large capacitance. Figure 2.26 shows two examples of flip-flop shifting.

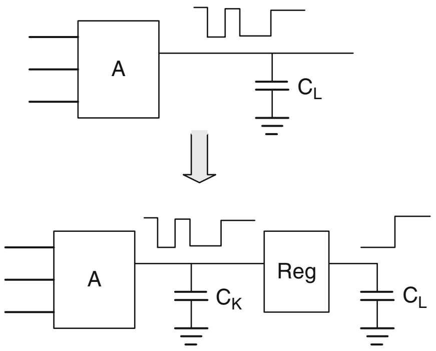


Fig. 2.25 Flip-flop insertion to minimize glitches

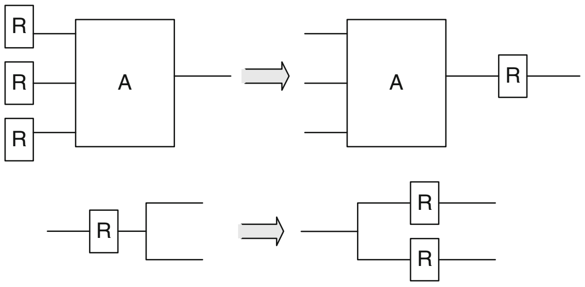


Fig. 2.26 Flip-flop shifting to reduce glitches

2.4.4 Circuit Level Optimization

In this level, circuit optimization and transistor sizing can be used to minimize power dissipation. We will first consider the circuit optimization techniques, such as equivalent pin reordering.

Equivalent pin reordering: Considering the NAND gate shown in Fig. 2.27, internal parasitic capacitance exists at the internal nodes. Suppose that the inputs of the NAND gate switch from 10 to 00. Before transition, the inputs can be

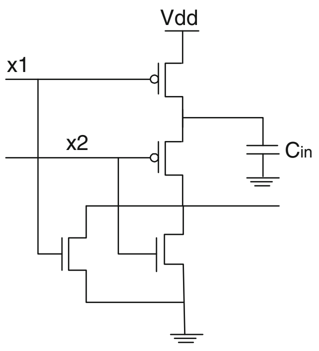


Fig. 2.27 NAND gate with internal parasitic capacitance

$x1 = 0, x2 = 1$ or $x1 = 1, x2 = 0$. Suppose $x1 = 0$ and $x2$ transfers from 1 to 0, because the first pMOS is already charged, only C_l needs charging during the transaction. However, if $x2 = 0$, and $x1$ transfers from 1 to 0, both C_l and C_{in} need to be charged. We can see that the second input assignment will consume more dynamic power during the transaction. From this example, it can be observed that if a signal switches more, it should be assigned closer to the output in order to save on power consumption.

2.4.4.1 Transistor Sizing

Another way to minimize power in circuit level is to perform transistor sizing. Transistors in the circuit can be sized to optimize for delay and power. To improve the switching speed of a particular block on the critical path, one can increase the widths of the transistors in the block. This provides increased driving current and better output transition time, and hence, smaller short-circuit power dissipation. However, it needs to be noted that increasing the transistor widths of the block also increase the loading capacitance of its preceding block, and may adversely affect the delay and short-circuit power. Thus, the issues regarding delay and power dissipation are very interlinked and so require careful analysis. On the other hand, an alternative method is to reduce the transistor size of the block in non-critical paths. In such a case, the transistor can be sized to reduce power consumption, while its increased delay is still less than the slack allowed before sizing. Hence, in this method, power reduction is achieved without affecting delay. Transistor sizing can reduce power to 5–10% of total power.

2.4.5 Power Gating

One side effect of IC dynamic power optimization is the potential increase of IC leakage. With each technology generation, circuit supply voltages must decrease accordingly to minimize IC dynamic power consumption. This unfortunately has negative impact on circuit performance. More specifically, circuit frequency, f, can be expressed in terms of supply voltage, V_{dd}, and threshold voltage V_{t}, as follows:

$$f = \frac{k(V_{\text{dd}} - V_{\text{t}})^{\alpha}}{V_{\text{dd}}} \tag{2.32}$$

where $1 \leq \alpha \leq 2$.

The above equation shows that f decreases as V_{dd} decreases. To address circuit performance concerns, it is then important to reduce the circuit threshold voltage V_{t} accordingly, which, however, results in exponential increase in IC leakage power dissipation.

Leakage power optimization has been intensively studied, and a variety of techniques have been proposed. Among which, power gating has been one of the most effective techniques, and thus is widely adopted in low-power IC design [93–95]. Power gating minimizes IC leakage power consumption by shutting off the power

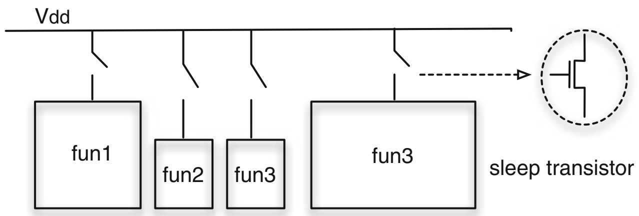


Fig. 2.28 Power gating for IC leakage power minimization

supply when a circuit is idle. As shown in Fig. 2.28, power gating implementations typically involve multi-threshold circuit designs. Circuit logic functions are implemented using high-performance, hence low V_t, technology, but are connected to the power supply through high V_t footer and/or header sleep transistors. During run-time circuit operation, idle functional units are identified, and the corresponding power supply is then shut off by turning off the sleep transistor. Since the the sleep transistor uses high V_t technology, the overall leakage current of the idle functional blocks can be substantially reduced.

However, power gating also introduces a number of design challenges. First, the design complexity and area overhead of power gating must be carefully considered. The area overhead is mainly introduced by sleep transistors and their corresponding control logic. In particular, due to stringent circuit performance requirements, sleep transistors have large footprints. In addition, to prevent floating output when a functional unit is in sleep mode, extra logic, such as weak pull-up/down devices, are required. Furthermore, ground bounce may be introduced during circuit power state transition due to parasitic capacitance charge and discharge, which has serious impact on circuit run-time performance and reliability. Addressing this problem requires careful design and placement of sleep transistors, as well as intelligent run-time wakeup scheduling. Overall, power gating is beneficial, but the design complexity and cost must be carefully considered.

2.5 Conclusions

This chapter overviewed the IC power consumption issue. It first explained the primary circuit power dissipation mechanisms, i.e., dynamic and leakage power consumption. It then described how to model and estimate IC power consumption. Finally, a wide range of recently proposed power optimization techniques were discussed. The following sections will provide more detailed discussion of various related IC power consumption issues, as well as recently proposed power management and optimization techniques. Even though low-power IC design has been an active research field over two decades, IC power consumption challenges will continue to grow, and power optimization will continue to be the one of the formost IC design objectives.

References

1. International Technology Roadmap for Semiconductors, http://public.itrs.net/, United States Semiconductor Industry Association, USA, 2007
2. Yuan CP, Trick TN (1982) A simple formula for the estimation of the capacitance of twodimensional interconnects in VLSI circuits. IEEE Electron Device Let EDL-3(12):391–393
3. Lee M (1996) A fringing and coupling interconnect line capacitance model for VLSI on-chip wiring delay and crosstalk. In: IEEE international symposium on circuits and systems, Atlanta, GA, USA, pp 233–236
4. Shoji M (1988) CMOS digital circuit technology. Prentice Hall, Englewood Cliffs, NJ
5. Vanoostende P et al (1992) Evaluation of the limitations of the simple CMOS power estimation formula: comparison with accurate estimation. In: Proceedings of European workshop on power and timing modelling, Paris, France, pp 16–25
6. Bakoglu H (1987) Circuits, interconnections, and packaging for VLSI. Addison-Wesley, New York, NY
7. Pullela S, Menezes N, Omar J, Pillage L (1993) Skew and delay optimization for reliable buffered clock trees. In: Digest of technical papers of IEEE international conference on computer aided design, Santa Clara, CA, USA, pp 556–562
8. Zhu D, Dai W, Xi J (Nov 1993) Optimal sizing of high speed clock networks based on distributed RC and transmission line models. In: Digest of technical papers of IEEE international conference on computer aided design, Santa Clara, CA, USA, pp 628–633
9. Veendrick HJM (1984) Short-circuit dissipation of static CMOS circuitry and its impact on the design of buffer circuits. IEEE JSSC 19(4):468–473
10. Rabaey JM, Pedram M (1996) Low power design methodologies. Kluwer, Boston, MA
11. Bisdounis L, Koufopavlou O (1999) Analytical modeling of short-circuit energy dissipation in submicron CMOS structures. In: Proceedings of IEEE international conference on electronics, circuits and systems, Pafos, Cyprus, pp 1667–1670
12. da Costa EAC, Cardoso R, Bampi S (2000) Modeling of short circuit power consumption using timing-only logic cell macromodels. In: Proceedings 13th symposium on integrated circuits and systems design, Manaus, Brazil, pp 222–227
13. Naffziger S et al (Jan. 2006) The implementation of a 2-core, multi-threaded Itanium family processor. IEEE Solid J-State Circuits 41(1):197–209
14. Mistry K, Allen C, Auth C, Beattie B, Bergstrom D, Bost M, Brazier M, Buehler M, Cappellani A, Chau R, Choi C-H, Ding G, Fischer K, Ghani T, Grover R, Han W, Hanken D, Hattendorf M, He J, Hicks J, Heussner R, Ingerly D, Jain P, James R, Jong L, Joshi S, Kenyon C, Kuhn K, Lee K, Liu H, Maiz J, McIntyre B, Moon P, Neirynck J, Pae S, Parker C, Parsons D, Prasad C, Pipes L, Prince M, Ranade P, Reynolds T, Sandford J, Shifren L, Sebastian J, Simon D, Sivakumar S, Smith P, Thomas C, Troeger T, Vandervoorn P, Williams S, Zawadzki K (Dec 2007) A 45 nm logic technology with High-k+Metal gate transistors, strained silicon, 9 Cu interconnect layers, 193 nm dry patterning, and 100% Pb-free packaging. In: IEEE international electron devices meeting, Washington, DC, USA
15. Roy K, Mukhopadhyay S, Mahmoodi-Meimand H (Feb 2003) Leakage current mechanisms and leakage reduction techniques in deep-submicrometer CMOS circuits. Proc IEEE 91(2):305–327
16. Chandrakasan A, Bowhill W, Fox F (2001) Design of high-performance microprocessor circuits. IEEE, Piscataway, NJ
17. Cao KM et al (Dec 2000) BSIM4 gate leakage model including source-drain partition. In: International Electron Device Meeting (IEDM) technology digest, San Francisco, CA, USA, pp 815–818
18. Nagata M (1992) Limitations, innovations and challenges of circuits and devices into a half micrometer and beyong. IEEE J Solid-State Circuits 27:465–472
19. Hu C (Jun 1994) MOSFET scaling in the next decade and deyong. In: Proceedings of international semiconductor device research symposium, Charlottesville, VA, USA, pp 105–114

20. Meindl JD (1995) Low power microelectronics: retrospect and prospect. Proc IEEE 83(4):619–635
21. Iwai H (1998) CMOS scaling towards its limits. In Proceedings of Solid-State and Integrated Circuit Technology, Beijing, China, pp 31–34
22. Frank DJ (Mar/May 2002) Power-constrained CMOS scaling limits. IBM J Res Dev. 46(2/3):235–244
23. Lundstrom M (Dec 2003) Device physics at the scaling limit: what matters? MOSFETs. In: Proc. IEEE Int. Electron Devices Meeting, Washington, DC, USA, pp 33.1.1–33.1.4
24. Nowak EJ, Aller I, Ludwig T, Kim K, Joshi RV, Chuang CT, Bernstein K, Puri R (Jan-Feb 2004) Turning silicon on its edge. IEEE Circuits Devices Mag 20(1):20–31
25. Mukhopadhyay S, Roy K (Aug 2003) Modeling and estimation of total leakage current in nano-scaled CMOS devices considering the effect of parameter variation. In: International symposium on low power electronics and design, Seoul, Korea, pp 172–175
26. Srivastava A, Bai R, Blaauw D, Sylvester D (2002) Modeling and analysis of leakage power considering within-die process variations. In: Int. symp. low power electronics and design, Monterey, CA, USA, pp 64–67
27. Rao R, Srivastava A, Blaauw D, Sylvester D (2003) Statistical estimation of leakage current considering inter- and intra-die process variation. In: International symposium on low power electronics and design, Seoul, Korea, pp 19–23
28. Narendra S, De V, Borkar S, Antoniadis D, Chandrakasan A (2002) Full-chip sub-threshold leakage power prediction model for sub-0.18um CMOS. In: International symposium on low power electronics and design, Monterey, CA, USA, pp 19–23
29. Agarwal K, Rao R, Sylvester D, Brown R (Jun 2007) Parametric yield analysis and optimization in leakage dominated technologies. Trans. VLSI Syst15(6):613–623
30. Rao R, Devgan A, Blaauw D, Sylvester D (2004) Parametric yield estimation considering leakage variability. Design automation conf., San Diego, CA, USA, pp 442–447
31. King SM (Oct 1986) Accurate simulation of power dissipation in VLSI circuits. IEEE J Solid State Circuits 21(5):889–891
32. Burch R, Najm FN, Yang P, Trick T (Mar 1993) A Monte Carlo approach for power estimation. IEEE Trans.VLSI syst 1(1):63–71
33. Goldstein H (Sept 1979) Controllability/observability of digital circuits. IEEE Trans. Circuits Syst 26(9):685–693
34. Lee C (Jul 1959) Representing of switching circuits by binary-decision diagrams. Bell Syst Tech J. 38:985–999
35. Bryant R (Aug 1986) Graph-based algorithms for Boolean function manipulation. IEEE Trans Comput Aided Des, C-35(8):677–691
36. Chou TL, Roy K, Prasad S (Nov 1994) Estimation of circuit activity considering signal correlations and simultaneous switching. In: Proceedings of IEEE international conference on computer-aided design, San Jose, CA, USA, pp 300–303
37. Ghosh A, Devadas S, Keutzer K, White J (Jun 1992) Estimation of average switching activity in combinational and sequential circuits. In: Proceedings of design automation conference, Anaheim, CA, USA, pp 253–259
38. Monteiro J, Devadas S, Ghosh A (Nov 1993) Retiming sequential circuits for low power. In: Proceedings of IEEE international conference on Computer Aided Design, Santa Clara, CA, USA, pp 398–402
39. Chen Z, Roy K (1998) A power macromodeling technique based on power sensitivity. In: Proceedings of design automation conference, San Francisco, CA, USA, pp 678–683
40. Gupta S, Najm FN (1997) Power macromodeling for high level power estimation. In: Proceedings of design automation conference, Anaheim, CA, USA, pp 365–370
41. Muttreja A, Raghunathan A, Ravi S, Jha NK (2004) Automated energy/performance macro-modeling of embedded software. In: Proceedings of design automation conference, pp 99–102
42. Butts JA, Sohi GS (Dec 2000) A static power model for architects. In: Proceedings of international symposium on microarchitecture, Monterey, CA, USA, pp 191–201

43. Martin SM et al (Nov 2002) Combined dynamic voltage scaling and adaptive body biasing for lower power microprocessors under dynamic workloads. In: Proceedings of international conference on computer-aided design, San Jose, CA, USA, pp 721–725
44. Narendra S et al (Feb 2004) Full-chip subthreshold leakage power prediction and reduction techniques for sub-0.18 CMOS. IEEE J Solid-State Circuits 39(2):501–510
45. Tsai YF et al (Nov 2004) Characterization and modeling of run-time techniques for leakage power reduction. IEEE Trans VLSI Syst 12(11):1221–1232
46. Abdollahi A, Fallah F, Pedram M (Feb. 2004) Leakage current reduction in CMOS VLSI circuits by input vector control. IEEE Trans VLSI Syst12(2):140–154
47. BSIM4. http://www-device.eecs.berkeley.edu/bsim3/bsim4.html, Berkeley University, CA, USA, 2009
48. HSPICE. http://www.synopsys.com/products/mixedsignal/hspice/hspice.html, Synopsys, CA, USA, 2005
49. Sirichotiyakul S et al (Apr 2002) Duet: an accurate leakage estimation and optimization tool for Dual- Vt circuits. IEEE Trans VLSI Syst10(2):79–90
50. Lee D et al (Jun 2003) Analysis and minimization techniques for total leakage considering gate oxide leakage. In Proceedings of design automation conference, Anaheim, CA, USA, pp 175–180
51. Rao RM et al (Aug 2003) Efficient techniques for gate leakage estimation. In: Proceedings of international symposium on low power electronics & design, Seoul, Korea, pp 100–103
52. Rastogi A et al (Jan 2007) An efficient technique for leakage current estimation in sub 65 nm scaled cmos circuits based on loading effect. In: Proceedings of international conference on, Bangalore, India, VLSI design
53. Do MQ et al (Mar 2007) Leakage-conscious architecture-level power estimation for partitioned and power-gated sram arrays. In: Proceedings of international symposium on quality of electronic design, San Jose, CA, USA, pp 185–191
54. Gopalakrishnan C, Katkoori S (Feb 2003) An architectural leakage power simulator for vhdl structural datapaths. In: Proceedings of international symposium on VLSI circuits, Tampa, FL, USA, pp 211–212
55. Kumar A, Anis M (Mar 2006) An analytical state dependent leakage power model for FPGAs. In: Proceedings of design automation and test in Europe conference, Munich, Germany, pp 612–617
56. Naveh A et al (May 2006) Power and thermal management in the Intel CoreDuo processor. Intel Technol J 10(2), DOI: 10.1535/itj.1002.03
57. Sato T et al (Jan 2005) On-chip thermal gradient analysis and temperature flattening for SoC design. In: Proceedings of the Asia and South Pacific design and automation conference, San Diego, CA, USA, pp 1074–1077
58. Lin S-C, Banerjee K (Nov 2006) An electrothermally-aware full-chip substrate temperature gradient evaluation methodology for leakage dominant technologies with implications for power estimation and hot-spot management. In: Proceedings of international conference on computer-aided design, San Jose, CA, USA, pp 568–574
59. Zhang Y et al (May 2003) HotLeakage: a temperature-aware model of subthreshold and gate leakage for architects. University of Virginia, Technical Report, CS-2003-05
60. Su H, Liu F, Devgan A, Acar E, Nassif S (Aug 2003) Full chip leakage estimation considering power supply and temperature variations. In: Proceedings of international symposium on low power electronics & design, Seoul, Korea, pp 78–83
61. Liao WP, He L, Lepak KM (July 2005) Temperature and supply voltage aware performance and power modeling at microarchitecture level. IEEE Trans Comput Aided Des Integr Circ Syst 24(7):1042–1053
62. Galaxy design platform. http://www.synopsys.com/products/solutions/galaxyplatform.html, Synopsys, CA, USA, 2008
63. Li P, Pileggi LT, Ashghi M, Chandra R (Nov 2004) Efficient full-chip thermal modeling and analysis. In: Proceedings of international conference on computer-aided design, San Jose, CA, USA, pp 319–326

64. Zhan Y, Sapatnekar SS (Oct 2005) A high efficiency full-chip thermal simulation algorithm. In: Proc. int. conf. computer-aided design, San Jose, CA, USA
65. Huang W, Ghosh S, Velusamy S, Sankaranarayanan K, Skadron K, Stan M (May 2006) HotSpot: A compact thermal modeling methodology for early-stage VLSI design. IEEE Trans VLSI Syst14(5):501–524
66. Smy T, Walkey D, Dew S (Jul 2001) Transient 3D heat flow analysis for integrated circuit devices using the transmission line matrix method on a quad tree mesh. Solid-State Electron 45(7):1137–1148
67. Liu P, Qi Z, Li H, Jin L, Wu W, Tan S, Yang J (Oct 2005) Fast thermal simulation for architecture level dynamic thermal management. In: Proceedings of international conference on computer-aided design, San Jose, CA, USA
68. Wang T, Chen C (Dec 2002) 3-D thermal-ADI: a linear-time chip level transient thermal simulator. IEEE Trans Comput Aided Des Integr Circ Syst 21(12):1434–1445
69. Yang Y, Zhu C, Gu ZP, Shang L, Dick RP (Nov 2006) Adaptive multi-domain thermal modeling and analysis for integrated circuit synthesis and design. In: Proceedings of international conference on computer-aided design, San Jose, CA, USA
70. Murthy JY, Narumanchi SVJ, Pascual-Gutierrez JA, Wang T, Ni C, Mathur SR (2005) Review of multi-scale simulation in sub-micron heat transfer. Int J Multiscale Comput Eng 3:5–32
71. Allec N, Hassan Z, Shang L, Dick RP, Yang R (Nov 2008) ThermalScope: multi-scale thermal analysis for nanometer-scale integrated circuits. In: Proceedings of international conference on computer-aided design, San Jose, CA, USA, pp 603–610
72. Raghunathan A, Jha NK (May 1995) An ILP formulation for low power based on minimizing switched capacitance during datapath allocation. In: Proceedings of international symposium on circuits & system, Seattle, WA, USA, pp 1069–1073
73. MeLna R, Rabaey J (Apr 1994) Behavioral level power estimation and exploration. In: Proc. Int. wkshp. low power design, Napa Valley, CA, USA, pp 197–202
74. Chaudhuri S, Blythe SA, Walker RA (Sept 1995) An exact solution methodology for scheduling in a 3D design space. In: Proceedings of international symposium on system level synthesis, Cannes, France, pp 78–83
75. Wang Q, Vrudhula SBK (1996) Multi-level logic optimization for low power using local logic transformations. In: Proceedings of IEEE international conference on computer-aided design, San Jose, CA, USA, pp 270–277
76. Irnan S, Pedram M (1994) Multi-level network optimization for low power. In: Proceedings of IEEE international conference on computer-aided design, San Jose, CA, USA, pp 372–377
77. SIS. http://embedded.eecs.berkeley.edu/pubs/downloads/sis/index.htm, Berkeley University, CA, USA, 2003
78. Roy K, Prasad SC (Dec 1993) Circuit activity based logic synthesis for low power reliable operations. IEEE Trans VLSI syst1(4):503–513
79. Keutzer K (Jun 1987) DAGON: technology mapping and local optimization. In: Proceedings of design automation conference, Miami Beach, FL, USA, pp 341–347
80. Tsui C-Y, Pedram M, Despain AM (1993) Technology decomposition and mapping targeting low power dissipation. In: Proceedings of design automation conference, Dallas, Texas, USA, pp 68–73
81. Tiwari V, Ashar P, Malik S (Jun 1993) Technology mapping for low power. In: Proceedings of design automation conference, Dallas, Texas, USA, pp 74–79
82. Lin B, De man H (Oct 1993) Low-Power driven technology mapping under timing constraints. In: Proceedings of international conference on computer design, Cambridge, MA, USA, pp 421–427
83. Wang C-C, Kwan C-P (1997) Low power technology mapping by hiding hightransition paths in invisible edges for LUT-based FPGAs. In: Proceedings of international symposium on circuits and systems, Hong Kong, pp 1536–1539

84. C-Y. Tsui, Pedram M, Despain AM (Nov 1993) Efficient estimation of dynamic power dissipation under a real delay model. In: Proceedings of IEEE international conference on computer-aided design, Santa Clara, CA, pp 224–228
85. C-Y. Tsui, Pedram M, C-H. Chen, Despain AM (Nov 1994) Low power state assignment targeting two- and multi-level logic implementation. In: Proceedings of IEEE international conference on computer-aided design, San Jose, CA, USA, pp 82–87
86. Olson E, Kang SM (1994) State assignment for low-power FSM synthesis using genetic local search. In: Proceedings of IEEE custom integrated circuits conference, San Diego, CA, pp 140–143
87. Venkataraman G, Reddy SM, Pomeranz I (2003) GALLOP: genetic algorithm based low power FSM synthesis by simultaneous partitioning and state assignment. In: Proceedings of international conference on VLSI design, New Delhi, India, pp 533–538
88. Chattopadhyay S, Reddy PN (2004) Finite state machine state assignment targeting low power consumption. Proc IEE Comput Dig Tech 151(1):61–70
89. Villa T, Sangiovanni-Vincentelli A (Sept 1990) NOVA: state assignment of finite state machines for optimal two-level logic implementations. IEEE Trans Comput Aided Des Integr Circ Syst 9:905–924
90. Lin B, Newton AR (Aug1989) Synthesis of multiple-level logic from symbolic high-level description languages. In: Proceedings of IFIP international conference on VLSI, Munich, Germany, pp 187–196
91. Narayanan U, Pan P, Liu CL (1998) Low power logic synthesis under a general delay model. In: Proceedings of international symposium on low power electronics and design, Monterey, CA, USA, pp 209–214
92. Hsu Y-L, Wang S-J (2002) Retiming-based logic synthesis for low-power. In: Proceedings of international symposium on low power electronics and design, Monterey, CA, USA, pp 275–278
93. Jiang H, Marek-Sadowska M, Nassif SR (Oct 2005) Benefits and costs of power-gating technique. In: Proceedings of international conference on computer design, San Jose, CA, USA, pp 559–566
94. Hu Z, Buyuktosunoglu A, Srinivasan V, Zyuban V, Jacobson H, Bose P (Aug 2004) Microarchitectural techniques for power gating of execution units. In: Proceedings of international symposium on low power electronics and design, Newport Beach, CA, USA, pp 32–37
95. Shi K, Howard D (Jul 2006) Challenges in sleep transistor design and implementation in lowpower designs. In: Proceedings of design automation conference, San Francisco, CA, USA, pp 113–116