Chapter 11
Ultra Low Power/Voltage Design

Slide 11.1
In previous chapters, we had established that considering energy in isolation rarely makes sense. Most often, a multi-dimensional optimization process is essential equally considering other metrics such as throughput, area, or reliability. Yet, in a number of applications, minimizing energy (or power) is the single most important goal, and all other measures are secondary. Under such conditions, it is worth exploring what the minimum energy is to perform a given task in a given technology. Another interesting question is whether and how this minimum changes with further scaling of the technology. Addressing these questions is the main goal of this chapter.

Slide 11.2
The chapter commences with the rationale behind the concept of ultra low power (ULP) design, and the establishment of some firm lower bounds on the minimum energy for a digital operation. From this, it emerges that ULP is quite synonymous to ultra low voltage (ULV) design. Unless we find a way to scale down threshold voltages without dramatically increasing the leakage currents, ULV circuits, by necessity, operate in the sub-threshold region. A sizable fraction of the chapter is hence devoted to the modeling, operation, and optimization of digital logic and memory operating in this mode. We will show that, though this most often leads to a minimum-energy design, it also comes at an...
exponentially increasing cost in performance. Hence, backing off just a bit to the moderate-inversion region gives almost identical results in energy, but with a dramatically better performance. An E–D optimization framework that covers all the possible operational regions of a MOS transistor (strong, moderate, and weak inversion) is proposed. Finally, we ponder the question whether other logic families than traditional complementary CMOS might not be better suited to enable ULP at better performance.

**Slide 11.3**

Already in the introductory chapter of this book, it had become apparent that the continuation of technology scaling requires the power density (i.e., the power consumed per unit area) to be constant. The ITRS projects a different trajectory though, with both dynamic and static power densities continuing to increase over the coming decade unless some innovative solutions emerge. One option is to have the circuit do less—reduce the activity, in other words—but this seems hardly attractive. The other more sensible approach is to try continuing the scaling of the energy per operation (EOP). This begs for an answer to the following questions: Is there an absolute lower bound on the EOP? And how far away are we from reaching it?

It turns out that answers to these queries are closely related to the question of the minimum supply voltage at which a digital logic gate can still operate, which turns out to be well-defined. The major topic of this chapter is the exploration of circuit techniques that allow us to approach as close to that minimum as possible.

**Slide 11.4**

Although keeping the power density constant is one motivation for the continued search to lower the EOP, another, maybe even more important, reason is the exciting applications that only become feasible at very low energy/power levels. Consider, for instance, the digital wristwatch. The concept, though straightforward, only became attractive once the power dissipation...
was made low enough for a single small battery to last for many years. As such, wristwatches in the early 1980s became the very first application using ultra low power and voltage design technologies.

Today, ULP technology is making it possible for a range of far more complex applications to become reality. Wireless sensor network nodes, combining integrated wireless front ends with signal acquisition and processing, are currently making their way into the market. Further power reductions by one or two orders of magnitude may enable even more futuristic functionality, such as intelligent materials, smart objects that respond to a much broader range of input sense, and the in situ observation of human cells. Each of these requires that the electronics are completely embedded into the object, and operate solely off the ambient energy. To realize this lofty goal, it is essential that power levels for the complete node are at the microwatt level, or below (remember the microwatt nodes described in Chapter 1).

### Minimum Operational Voltage of Inverter

- Swanson, Meindl (April 1972)
- Further extended in Meindl (Oct 2000)

**Limitation: gain at midpoint > -1**

\[
V_{fo,\text{min}} = 2\left(\frac{kT}{q}\right)\ln(2 + \frac{C_l}{C_{ox}})
\]

or

\[
V_{fo,\text{min}} = 2\left(\frac{kT}{q}\right)\ln(1 + n)
\]

- \(C_l\): gate capacitance
- \(C_{ox}\): diffusion capacitance
- \(n\): slope factor

For ideal MOSFET (60 mV/decade slope):

\[
V_{on,\text{min}} = 2\ln(2)\frac{kT}{q} = 1.38\frac{kT}{q} = 0.036 \text{ V}
\]

at a temperature of 300 K

[Ref: R. Swanson, JSSC’72; J. Meindl, JSSC’00]

---

**Slide 11.5**

The question of the minimum operational voltage of a CMOS inverter was addressed in a landmark paper [Swanson72] in the early 1970s – published even before CMOS integrated circuits came in vogue! For an inverter to be regenerative and to have two distinct steady-state operation points (a “1” and a “0”), it is essential that the absolute value of the gain of the gate in the transient region be larger than 1. Solving for those conditions leads to an expression for \(V_{\text{min}}\) equal to \(2(kT/q)\ln(1 + n)\), where \(n\) is the slope factor of the transistors. One important observation is that \(V_{\text{min}}\) is proportional to the operational temperature \(T\). Cooling down a CMOS circuit to temperatures close to absolute zero (e.g., liquid Helium), makes operation at mV levels possible. (Unfortunately, the energy going into the cooling more than often offsets the gains in operational energy.) Also, the closer the MOS transistor operating in sub-threshold mode gets to the ideal bipolar transistor behavior, the lower the minimum voltage. At room temperature, an ideal CMOS inverter (with a slope factor of 1) could marginally operate at as low as 36 mV!
Sub-threshold Modeling of CMOS Inverter

- From Chapter 2:

\[
I_{DS} = I_s e^{\frac{V_{DS} - V_{TH}}{kT/q}} \left( 1 - e^{\frac{-V_{DS}}{kT/q}} \right) = I_s e^{\frac{V_{TH}}{kT/q}} \left( e^{\frac{V_{DS}}{kT/q}} - 1 \right)
\]

where

\[
I_0 = I_s e^{\frac{-V_{TH}}{kT/q}}
\]

(DIBL can be ignored at low voltages)

of clarity. For low values of \( V_{DS} \), the DIBL effect can be ignored.

Sub-threshold DC model of CMOS Inverter

Assume NMOS and PMOS are fully symmetrical and all voltages normalized to the thermal voltage \( \Phi_T = kT/q \)

\( x_i = V_i/\Phi_T; \ x_n = V_n/\Phi_T; \ x_D = V_D/\Phi_T \)

The VTC of the inverter for NMOS and PMOS in sub-threshold can be derived:

\[
x_i = x_n + \ln \left( 1 + \sqrt{\frac{(G-1)^2 + 4Ge^{\frac{-x_n}{G}}}{2}} \right)
\]

where \( G = e^{\frac{V}{\Phi_T}} \)

so that

\[
A_v = \frac{2(1 - e^{\frac{-x_n}{G}} - e^{\frac{-x_n}{G}})}{n(2e^{\frac{-x_n}{G}} - e^{\frac{-x_n}{G}} - e^{\frac{-x_n}{G}})} \quad \text{and} \quad A_{v_{MAX}} = -\left( e^{\frac{V}{\Phi_T}} - 1 \right) / n
\]

For \( |A_{v_{MAX}}| = 1 \): \( x_D = 2\ln(n+1) \)  

[Ref: E. Vittoz, CRC/05]

Slide 11.6

Given the importance of this expression, a quick derivation is worth undertaking. We assume that at these low operational voltages, the transistors operate only in the sub-threshold regime, which is often also called the weak-inversion mode. The current–voltage relationship for a MOS transistor in sub-threshold mode was presented in Chapter 2, and is repeated here for the sake of clarity. For low values of \( V_{DS} \), the DIBL effect can be ignored.

Slide 11.7

The (static) voltage transfer characteristic (VTC) of the inverter is derived by equating the current through the NMOS and PMOS transistors. The derivation is substantially simplified if we assume that two devices have exactly the same strength when operating in sub-threshold. Also, normalizing all voltages with respect to the thermal voltage \( \Phi_T \) leads to more elegant expressions. Setting the gain to \(-1\) yields the same expression for the minimum voltage as was derived by Swanson.

Slide 11.8

Using the analytical models derived in the previous slide, we can plot the VTC of the inverter. It becomes clear that, when the normalized supply voltage approaches its minimum value, the VTC degrades, and the static noise margins are reduced to zero. With no gain in the intermediate region, distinguishing between “0” and “1” becomes impossible, and a flip-flop composed of such inverters would no longer be bi-stable. This presents a boundary condition. For reliable operation, a margin must be provided. As can be seen from the plots, setting the supply voltage at 4 times the
thermal voltage leads to reasonable noise margins (assuming $n = 1.5$). This is approximately equal to 100 mV.

Slide 11.9
Simulations (for a 90 nm technology) confirm these results. When plotting the minimum supply voltage as a function of the PMOS/NMOS ratio, a minimum can be observed when the inverter is completely symmetrical, that is when the PMOS and NMOS transistors have identical drive strengths. Any deviation from the symmetry causes $V_{\text{min}}$ to rise. This implies that transistor sizing will play a role in the design of minimum-voltage circuits.

Also worth noticing is that the simulated minimum voltage of 60 mV is slightly higher than the theoretical value of 48 mV. This is mostly owing to the definition of “operational” point. At 48 mV, the inverter is only marginally functional. In the simulation, we assume a small margin of approximately 25%.

Slide 11.10
The condition of symmetry between pull-up and pull-down networks for minimum-voltage operation proves to be important in more complex logic gates. Consider, for instance, a two-input NOR gate. The drive strengths of the pull-up and pull-down networks depend upon the input values.
Also Holds for More Complex Gates

Minimum operational supply voltage
(two-input NOR)

Degradation due to asymmetry

When only one input is switched (and the other fixed to “0”), the built-in asymmetry leads to a higher minimum voltage than when both are switched simultaneously. This leads to a useful design rule-of-thumb: when designing logic networks for minimum-voltage operation, one should strive to make the gate topology symmetrical over all input conditions.

Minimum Energy per Operation

Predicted by von Neumann: \( kT \ln(2) \)

J. von Neumann,
[Theory of Self-Reproducing Automata, 1966]

- Moving one electron over \( V_{Dmin} \):
  - \( E_{min} = qV_{Dmin} = q2(\ln(2))kT \approx qkT \ln(2) \)
  - Also called the Von Neumann–Landauer–Shannon bound
  - At room temperature (300 K): \( E_{min} \approx 0.29 \times 10^{-19} \text{J} \)

- Minimum sized CMOS inverter at 90 nm operating at 1V
  - \( E = CV_q^2 = 0.8 \times 10^{-19} \text{J} \), or 5 orders of magnitude larger!

How close can one get?
[Ref. J. Von Neumann, III’66]

Slide 11.11

Now that the issue of the minimum voltage is settled, the question of the minimum energy per operation (EOP) can be tackled. In a follow-up to the 1972 paper by Swanson, Meindl [Meindl, JSSC’00] argued that moving a single electron over the minimum voltage requires an energy equal to \( kT \ln(2) \). This result is remarkable in a number of ways.

- This expression for the minimum energy for a digital operation was already predicted much earlier by John von Neumann (as reported in [von Neumann, 1966]). Landauer later established that this is only the case for “logically irreversible” operations in a physical computer that dissipate energy by generating a corresponding amount of entropy for each bit of information that then gets irreversibly erased. This bound hence does not hold for reversible computers (if such could be built) [Landauer, 1961].

- This is also exactly the same expression that was obtained in Chapter 6 for the minimum energy it takes to transmit a bit over an interconnect medium. That result was derived from Shannon’s theorem, and was based on information-theoretic arguments.

The fact that the same expression is obtained coming from a number of different directions seems to be surprising at first. Upon closer analysis, all the derivations are based on a common
assumption of white thermal noise with a Gaussian distribution. Under such conditions, for a signal to be distinguishable it has to be at a level of \( \ln(2) \) above the noise floor of \( kT \). In light of its many origins, \( kT \ln(2) \) is often called the Shannon–von Neumann–Landauer limit.

A more practical perspective is that a standard 90 nm CMOS inverter (with a 1 V supply) operates at an energy level that is approximately five orders of magnitude higher than the absolute minimum. Given that a margin of 100 above the absolute lower bound is probably necessary for reliable operation, this means that a further reduction in EOP by three orders of magnitude may be possible.

\[
\begin{align*}
  t_p &= \frac{C V_{DD}}{I_{in}} = \frac{C V_{DD}}{V_{DD} I_0 e^{x_d} \Phi_T} \quad \text{(for} \ V_{DD} \gg \Phi_T \text{)} \\
  \text{Normalizing} \ t_p \ \text{to} \ \tau_0 = C \Phi_T / I_0: \quad \tau_p &= \frac{t_p}{\tau_0} = x_d e^{-x_d/n} \\
  \text{Comparison between curve-fitted model and simulations (FO4, 90 nm)}
\end{align*}
\]

Slide 11.12

The above analysis, though useful in setting absolute bounds, ignores some practical aspects, such as leakage. Hence, lowering the voltage as low as we can may not necessarily be the right answer to minimize energy.

Operating an inverter in the sub-threshold region, however, may be one way to get closer to the minimum-energy bound. Yet, as should be no surprise, this comes at a substantial cost in performance. Following the common practice of this book, we again map the design task as an optimization problem in the E–D space.

One interesting by-product of operating in the sub-threshold region is that the equations are quite simple and are exponentials (as used to be the case for bipolar transistors). Under the earlier assumptions of symmetry, an expression of the inverter delay is readily derived. Observe again that a reduction in supply voltage has an exponential effect on the delay!
Slide 11.13
Given the very low current levels, it should be expected that waveforms would exhibit very slow rise and fall times. Hence, the impact of the input slope on delay might be considerable. Simulations show that the delay indeed rises with the input transition time, but the impact is not unlike what we observed for above-threshold circuits. The question also arises whether short-circuit currents (which were not considered in the expression derived in Slide 11.12) should be included in the delay analysis. This turns out to be not the case as long as the input and output signal slopes are balanced, or the input slope is smaller than $t_0$.

Slide 11.14
An expression for the power dissipation is also readily derived. For $x_d \geq 4$, the logic levels are approximately equal to the supply rails, and the leakage current simply equals $I_0$. For dynamic and static leakage power to be combined in a single expression, an activity factor $\alpha$ ($= 2t_p/T$) should be introduced, just as we did in Chapter 3.
Power–Delay Product and Energy–Delay

\[ \text{pdp} = \frac{P_\text{d}}{C \Phi_T^2} = x_\delta \left( \frac{\alpha}{2} + e^{-\alpha/\alpha} \right) \]

\[ \text{ed} = \left( \frac{P_\text{d}}{C \Phi_T^2 / \pi} \right) = x_\delta \left( \frac{\alpha}{2} + e^{-\alpha/\alpha} \right) \]

For low activity (\( \alpha \ll 1 \)), large \( x_\delta \) advantageous!

**Slide 11.15**

We can now plot the normalized power–delay and energy–delay curves as a function of the normalized supply voltage \( x_\delta \) and the activity \( \alpha \). When the activity is very high (\( \alpha \) close to 1), the dynamic power dominates, and keeping \( x_\delta \) as low as possible helps. On the other hand, if the activity is low, increasing the supply voltage actually helps, as it decreases the clock period (and hence the amount of energy, leaking away over a single period).

**Slide 11.16**

The power–delay and energy–delay metrics, as we have indicated earlier, are somewhat meaningless, as they fail to take the energy–delay trade-off into account. A more relevant question is what voltage minimizes the energy for a given task and a given performance requirement. The answer is simple: keep the circuit as active as possible, and reduce the supply voltage to the minimum allowable value. Low-activity circuits must be operated at higher voltages, which leads to higher energy per operation. This opens the door for some interesting logic- and architecture-level optimizations. Shallow logic is preferable over long complex critical paths. Similarly, increasing the activity using, for instance, time multiplexing is a good idea if minimum energy is the goal.

**Slide 11.17**

To demonstrate the potential of sub-threshold design and its potential pitfalls, we analyze a couple of case studies. In the first example, an energy-aware Fast Fourier Transform (FFT) module [A. Wang, ISSCC’04] is analyzed. As is typical for FFTs, the majority of the hardware is dedicated to the computation of the “butterfly” function, which involves a complex multiplication. Another
major component of the module is the memory block, storing the data vector. The architecture is parameterized, so that FFTs of lengths ranging from 128 to 1024 points, and data word lengths from 8 to 16 bit can be computed efficiently.

---

Slide 11.18
Using models in the style we have been doing so far, the energy–delay space of the FFT module is explored. In this study, we are most interested in determining the minimum energy point and its location. The design parameters under consideration are the supply and threshold voltages (for a fixed circuit topology).

The simulated energy–delay plots provide some interesting insights (for a 180 nm technology):

- There is indeed a minimum energy point, situated well in the sub-threshold region ($V_{DD} = 0.35 \text{V}; V_{TH} = 0.45 \text{V}$).
- This operation point occurs at a clock frequency of just above 10 kHz – obviously not that stellar . . . but performance is not always an issue.
- As usual, an optimal energy–delay curve can be traced (shown in red).
- Lowering the throughput even further by going deeper into sub-threshold does not help, as the leakage power dominates everything, and causes the energy to increase.
- On the other hand, the performance can be increased dramatically (factor 10) if a small increase in energy above the minimum (25%) is allowed. This illustrates again that sub-threshold operation minimizes energy at a large performance cost.
Slide 11.19
The simulation data are confirmed by actual measurements from a 180 nm prototype chip. The design is shown to be functional from 900 mV down to 180 mV. Our earlier analysis would have indicated even lower operational supply voltages – however, as we made apparent in Chapters 7 and 9, SRAM memories tend to be the first points of failure, and probably are setting the lower bound on the supply voltage in this case.

From an energy perspective, simulations and measurements track each other quite well, confirming again the existence of a minimum-energy point in the sub-threshold region.

Slide 11.20
This case study, though confirming the feasibility of operation in the weak-inversion region, also serves to highlight some of the challenges. Careful sizing is essential if correct operation is to be guaranteed. Symmetrical gate structures are of the essence to avoid the impact of data dependencies. Deviations caused by process variations further complicate the design challenge – especially threshold variations have a severe impact in the sub-threshold regime. And as mentioned in the previous slide, ensuring reliable memory operation at these ultra low voltages is not a sinecure. In the following slides, we explore some of these design concerns.

Slide 11.21
The operation in the sub-threshold region boils down to the balancing of the PMOS and NMOS leakage currents against each other. If one of the devices is too strong, it overwhelms the other device under all operational conditions, resulting in faulty operation. For the case of a simple
Logic-Sizing Considerations

- CMOS in sub-threshold is “ratioed logic”
- Careful sizing of transistors necessary to ensure adequate logic levels

Inverter with a minimum-sized $W_n$

$W_n(\text{min})$, $W_n(\text{max})$

Max Size

Operational Region

Min Size

[Ref: A. Wang, ISSCC'04]

Slide 11.22

However, when process variations are taken into account, those margins shrink substantially. Simulations indicate that guaranteeing correct operation under all conditions is hard for supply voltages under 200 mV. From previous chapters, we know that the designer has a number of options to address this concern. One possibility is to use transistors with larger-than-minimum channel lengths. Unfortunately, this comes again with a performance penalty. Adaptive body biasing (ABB) is another option, and is quite effective if the variations are correlated over the area of interest.

Inverter and a given size of the NMOS transistor, the minimum and maximum widths of the PMOS transistor that bound the operational region can be computed, as shown in the graph. From a first glance, the margins seem to be quite comfortable unless the supply voltage drops below 100 mV.
Slide 11.23
The importance of maintaining symmetry under different data conditions is illustrated in this slide, where the functionality of two different XOR configurations is examined. The popular four-transistor XOR1 topology fails at low voltages for A = 1 and B = 0. The more complex transmission gate based XOR2 performs fine over all input patterns.

Slide 11.24
The reasons behind this failure become clear when the circuit is slightly redrawn. The failure of XOR1 for A = 1 and B = 0 is again caused by asymmetry: under these particular conditions, the leakage of the three “off” transistors overwhelms the single “on” PMOS device, resulting in an undefined output voltage. This is not the case for the XOR2, in which there are always two transistors pulling to VDD and GND, respectively.

Slide 11.25
We have now demonstrated that logic can be safely operated at supply voltages of around 200 mV. This begs the question whether the same holds for another essential component of most digital designs: memory. From our discussions on memory in previous chapters (Chapters 7 and 9), you can probably surmise that this may not be that easy. Scaling down the supply voltage reduces the read, write, and hold static noise margins (SNMs). In addition, it makes the memory more sensitive to process variations, soft errors and erratic failures. Leakage through the cell-access transistors negatively impacts the power.
When evaluating the potential impact of the different factors, it becomes apparent that the degradation of the read SNM is probably of primary concern. The bottom plot shows the distribution of the read and hold SNMs of a 90 nm 6T cell operated at 300 mV. As expected, the average value of the hold SNM (96 mV) is substantially higher than the read SNM (45 mV). In addition, the distribution is substantially wider, extending all the way to 0 mV.

**Slide 11.26**

This indicates that business as usual will not work for sub-threshold memory cells. One approach to combat the impact of variations is to increase the sizes of the transistors. Unfortunately, this leads, by necessity, to increased leakage, and may offset most of the power gains resulting from the voltage scaling.
Slide 11.27
This slide presents a bit-cell that has proven to operate reliably down to 300 mV. It specifically addresses the susceptibility to read-SNM failures at low supply voltages. By using a separate read-access buffer, the read process is addressed separately from the cell optimization. Hence, the supply voltage can be reduced without jeopardizing the read operation. Obviously, this comes at a cost of three extra transistors. This is only one possible cell topology. Other authors have proposed a number of similar approaches (e.g., [Chen’06]).

Slide 11.28
A 256 kb SRAM memory was designed and tested using the cell of the previous slide. Reliable operation down to 400 mV was demonstrated.

Though this represents great progress, it is clear that addressing further voltage scaling of SRAMs (or any other memory type) is essential if ultra low power design is to be successful. (sorry if this starts to sound like a broken record!)

Slide 11.29
Sub-threshold digital operation is also effective for the implementation of ultra low energy embedded microprocessors or microcontrollers. A sub-threshold processor for sensor network applications was developed by researchers at the University of Michigan. Implemented in a 130 nm CMOS technology with a 400 mV threshold (for $V_{DS} = 50$ mV), a minimum energy per instruction of 3.5 pJ is reached for a supply voltage of 350 mV. This is by far the lowest recorded energy efficiency for microprocessors or signal processors. At lower supply voltages, leakage power starts to dominate, and energy per instruction creeps upward again. If only the processor core is considered (ignoring memories and register files), the optimal-energy voltage shifts to lower levels.
time techniques only is hard. The use of runtime body-biasing of NMOS and PMOS transistors helps to compensate for the differences in driving strengths. In [Hanson'07], it was established that optimal application of substrate biasing helps to reduce the minimum operational voltage by 30–150 mV.

---

**Slide 11.30**

A prototype implementation of the sub-threshold processor is shown in this slide [Hanson07]. With the power dissipation of the processor low enough, it actually becomes possible to power the digital logic and memories from solar cells integrated on the same die. This probably represents one of the first ever totally energy self-contained processor dies. In addition to a number of processors, the chip also contains various test circuits as well as voltage regulators.

Note: Just recently, the same group published another incarnation of their ULP processor (now called the Phoenix) [Seok'08]. Implemented in a 180 nm CMOS process, the processor consumes only 29.6 pW in sleep mode, and 2.8 pJ/cycle in active mode at $V_{DD} = 0.5$ V for a clock frequency of 106 kHz. Observe that the authors deliberately chose to use an older-technology node to combat the leakage problems associated with newer processes. This turns out to be a big plus in reducing the standby power.
Is Sub-threshold the Way to Go?

- Achieves lowest possible energy dissipation
- But … at a dramatic cost in performance

Yet, you should be fully aware that sub-threshold design comes with an enormous performance penalty. This is apparent in this chart, which plots the propagation delay as a function of the supply voltage (on a linear scale). As we had established earlier, an exponential degradation of the delay occurs once the circuit drops into the sub-threshold region. For applications that happily operate at clock rates of 100s of kHz, this is not an issue. However, not many applications fall in this class.

In Addition: Huge Timing Variance

- Normalized timing variance increases dramatically with $V_{DD}$ reduction
- Design for yield means huge overhead at low voltages:
  - Worst-case design at 300 mV: >200% overkill

Slide 11.31
The preceding discussion clearly demonstrated that sub-threshold operation is a good way to go if operating at minimum energy is the primary design goal. Although this was well-known in the ultra low power design community for a while (dating back to the late 1970s), it has caught the attention of the broader design community only over the past few years. This is inspiring some very interesting efforts in academia and industry, which will be worth-tracking over the coming years.

Slide 11.32
In addition, the design becomes a lot more vulnerable and sensitive to process variations. This is clearly demonstrated in this chart, which plots the variance of the delay as a function of the supply voltage. When operating under worst-case design conditions, this results in a huge delay penalty.
Increased Sensitivity to Variations

- Sub-threshold circuits operate at low $I_{on}/I_{off}$ ratios, from about 1000 to less than 10 (at $x_d = 4$)
- Small variations in device parameters can have a large impact, and threaten the circuit operation

![Graph showing $I_{on}/I_{off}$ ratio vs. $x_d$]

Vulnerabilities (as we had demonstrated in the design of the FFT processor). Yet, variability clearly puts a limit on how far sub-threshold design can be driven.

---

**Slide 11.33**

Beyond the impact on performance, process variations may also threaten the reliability of the sub-threshold circuits. This is illustrated magnificently with a plot of the $I_{on}/I_{off}$ ratio of an inverter as a function of the supply voltage (using the expressions derived earlier in the chapter). With ratios in the small tens, variations can easily cause the circuit to fail, or force an increase in the supply voltage. Proper transistor sizing can help to mitigate some of these vulnerabilities.

---

**Slide 11.34**

A very attractive option to deliver both low energy as well as reasonable performance is obtained by practicing one of our established rules of thumb: that is, operating at the extremities of the energy–delay space is rarely worth it. Backing off just a bit gets you very close at a much smaller cost. Operating a circuit just above the threshold voltages avoids the exponential performance penalty.

One of the reasons designers avoided this region for a long time was the (mostly correct) perception that transistor models in this region are inaccurate. Manual analysis and optimization in this region is considered hard as well, as the simple performance models (such as the $V_T$-law) do not apply any longer. However, the combination of reducing supply and constant threshold voltages has made the moderate-inversion region increasingly attractive. Even better, transistor and performance models are now available that cover all the operational regions of the MOS transistor equally well.
Modeling over All Regions of Interest

- The EKV Model covers strong, moderate, and weak inversion regions

\[ I_s = k \frac{CV_{OD}}{IC \cdot I_S} \]

where \( k \) is a fit factor, \( I_s \) the specific current and \( IC \) the inversion coefficient.

- Inversion Coefficient \( IC \) measures the degree of saturation

\[ IC = \frac{I_{DS}}{I_S} = \frac{I_{DS}}{2n_{inf}C_0 \frac{V}{L} \Phi_r^2} \]

and is related directly to \( V_{DD} \)

\[ V_{DD} = V_{th} + 2n_{inf} \Phi_r \ln\left(e^{\frac{V}{V_{th}} - 1}\right) \]

[Ref: C. Enz, Analog '95]

A transistor operates in moderate inversion when \( IC \) is between 1 and 10. \( IC \) is a direct function of the supply and threshold voltages.

Relationship Between \( V_{DD} \) and \( IC \)

The EKV-based delay equation presented in Slide 11.33 provides an excellent match over a broad operation range (after fitting the model parameters \( k \) and \( I_S \) to the simulation data). Though the delay model performs very well in the weak- and moderate-inversion regions, some deviation can be observed for stronger inversion, caused by the fact that the model does not handle velocity saturation very well. However, this should not be an issue in most contemporary designs, in which

Slide 11.35

In 1995, Enz and Vittoz [Enz95] introduced the EKV MOS transistor model, which covers all operational regions of the transistor in a seamless manner. A crucial parameter in the model is the Inversion Coefficient \( IC \), which measures the degree of inversion of the device. Large values of \( IC \) indicate strong inversion, whereas values substantially below mean that the transistor operates in weak inversion (or sub-threshold). The

Slide 11.36

The relationship between supply voltage and inversion coefficient \( IC \) is plotted in this chart (for a given threshold voltage). Moving from weak to moderate and strong inversion requires an increasing amount of supply voltage investment.

Slide 11.37

The EKV-based delay equation presented in Slide 11.33 provides an excellent match over a broad operation range (after fitting the model parameters \( k \) and \( I_S \) to the simulation data). Though the delay model performs very well in the weak- and moderate-inversion regions, some deviation can be observed for stronger inversion, caused by the fact that the model does not handle velocity saturation very well. However, this should not be an issue in most contemporary designs, in which
the $IC$ rarely exceeds 100. Also, more recent incarnations of the model have resolved this issue. This analysis demonstrates that there should be no fear in operating a design at the boundary of the moderate and weak inversion.

Slide 11.38
An energy model based on $IC$ can be constructed as well. Again, the activity parameter $\alpha (= \frac{2t_p}{T})$ is used to combine the active and static energy components. The capacitance $C$ is the effective capacitance, which is the average switching capacitance per operation. The energy-per-operation (EOP) is plotted as a function of $IC$ and $\alpha$. The full lines represent the model, whereas the dots represent simulation results. As can be expected, the EOP is minimized for all operational regimes when $\alpha = 1$. Under that condition, reducing the supply voltage as much as possible and operating the circuit at very low values of $IC$ make sense. However, for less-active circuits, the minimum energy point shifts to larger values of $IC$, and eventually even moves to the moderate threshold region.

Slide 11.39
We can now combine the EKV delay and energy models, and plot the energy and delay planes as a function of $V_{DD}$ and $V_{TH}$, with activity as an independent parameter. For an activity level of 0.02, we see that the minimum EOP (red dots) moves from the weak- to the moderate-inversion region.
for increasing performance values. It is interesting to notice that the increased performance is obtained almost completely from a reduction of the threshold voltage at a fixed supply voltage. Also, observe how the equal-energy curves are almost horizontal, which illustrates how little impact the increased performance has on the EOP.

Slide 11.40
When the activity is further reduced ($\alpha = 0.002$), the minimum-energy point occurs at a higher supply voltage. Hence, the moderate-inversion region becomes attractive even earlier.

This set of simple-analysis examples shows that the EKV model is an effective tool in analyzing the energy–delay performance over a broad range of operation regimes. Even from this simple example, we can deduce some important guiding principles (most of which we had already mentioned when discussing sub-threshold behavior):

- From an energy perspective, a circuit performs better if the activity (i.e., duty cycle) is as high as possible.
- Sometimes this is not possible – for instance, if the required throughput is substantially below what the technology can deliver, the optimum operation point is not within the reliable
operation range of the technology, or the operation conditions vary over a wide range. In such cases, there are a number of possible solutions to consider:

1. Change the architecture to increase activity (e.g., time multiplexing).
2. Run the circuit at a high activity level, and put it in standby when finished.
3. Find the best possible operation condition given the lower activity level. This will often be at a higher supply voltage.

Slide 11.41
Whereas the example used in the previous slides is quite indicative in illustrating the potential of the EKV model, it is quite simple as well. To demonstrate that the modeling and optimization strategy also works for more complex examples, let us consider the case of a full adder cell. The cell we are considering is implemented using only two-input NAND gates and inverters.

The EKV model for the 90 nm CMOS technology-of-choice was derived in the earlier slides. The logical-effort model parameters are also available. For a small effort, we can now derive the delay and energy models of the complete adder.

Slide 11.42
The resulting energy–delay curves as a function of activity (and the inversion coefficient IC) are plotted here. Observe that these analytical results now span all operational regions. As expected, the largest energy savings are obtained when migrating from the strong-to the moderate-inversion region. Going from moderate to weak inversion has only a minor impact on the EOP. Evident also is that the lowest delay comes at a huge penalty in energy. In addition, the
graph shows that a larger reduction in energy consumption is achievable for lower activity factors. This can largely be explained by the increased importance of leakage currents at low activities.

**Slide 11.43**

Another advantage of having analytical delay and energy expressions available is that sensitivities to parameter variations can be easily computed. In this slide, we plot the normalized sensitivities of delay and energy with respect to variations in the supply and threshold voltages (for different values of $V_{TH}$). Please observe that these sensitivities hold only for small deviations of the corresponding variable, as they represent the gradients of energy and delay.

The plots on the left show a major increase in delay sensitivity with respect to both $V_{DD}$ and $V_{TH}$ when going into weak inversion (more or less independent of the chosen $V_{TH}$). The sensitivity of energy with respect to $V_{DD}$ is positive when dynamic energy dominates, and negative when leakage takes the largest share. This makes intuitive sense: an increase in $V_{DD}$ in strong inversion mainly affects the energy consumption; in weak inversion, it increases the performance substantially, reducing the time the circuit is leaking. The absolute sensitivity for energy with respect to $V_{TH}$ turns out to be equal for all values of $V_{TH}$. However, owing to the different absolute values of the energy for different $V_{TH}$, a maximum in the normalized sensitivity occurs at the point where the absolute energy is at its minimum.

**Slide 11.44**

So far, we have investigated how ultra low energy operation can be obtained by further lowering of the supply voltage, while keeping the threshold voltage more or less constant. Though this enables substantial energy reductions, the ultimate savings are limited. Moreover, the Von Neumann–Landauer–Shannon bound stays far out of reach. This becomes more than apparent in the graph on this slide, which shows the energy–delay curves for an inverter for technologies ranging from 90 nm to 22 nm. The graphs are obtained from simulations of a 423-stage ring-oscillator using the predictive technology models (PTM), operating at the nominal threshold voltages. Though it is clear from the simulations that the minimum EOP will continue to scale down, the projected reduction is quite small – no more than a factor of four, over five technology generations. This, of course, does not reflect potential technology innovations that are not included in the models. Even when assuming that the ultimate energy bound is a factor of 500 above $kT\ln(2)$ (for reliability reasons), the 22 nm solution still is at least a factor of 40 above that!

Hence, we cannot refrain from wondering what is needed to reduce the energy even further. Aside from adopting completely different technologies, it seems that the only plausible option is to
Moving the Minimum Energy Point

- Having the minimum-energy point in the sub-threshold region is unattractive
  - Sub-threshold energy savings are small and expensive
  - Further technology scaling not offering much relief
- Can it be moved upward?
- Or equivalently… Can we lower the threshold?

Remember the stack effect …

find ways to further reduce the threshold voltages while avoiding substantial increases in leakage current. The net effect of this would be to move the minimum-energy point out of weak into moderate inversion. A number of ways to accomplish this can be envisioned:

- **Adoption of switching devices with steep sub-threshold slopes** ( < 60 mV/dec) – Such transistors, if available, would definitely provide a major boost to energy-efficient logic. Some promising technologies have recently been proposed by the research community. Unfortunately, reliable production of any of those is still a long distance away. It is definitely worthwhile keeping an eye on the developments in this area, anyhow.

- **Adoption of different logic styles** – One of the main disadvantages of complementary CMOS logic is that leakage control is not easy. Other logic styles offer more precise control of the off-current, and hence may be better candidates for ultra low voltage/power logic.

Complex Versus Simple Gates

- Example (from Chapter 4)

  ![Diagram](image)

  **Complex gates improve the $I_{on}/I_{off}$ ratio!**

In Chapter 4, we analyzed the *stack effect*, which causes the off-current to grow slower than the on-current as a function of fan-in in a cascode connection of transistors. Exploiting this effect may help to create logic structures with reasonable $I_{on}/I_{off}$ ratios, even while reducing the threshold voltages. The minimum-energy point for large fan-in (complex) gates hence should be located at lower threshold voltages. It should be repeated that, as an additional benefit, complex gates typically come with lower parasitic capacitances.
Slide 11.46
The simulations shown in this slide, in which the equal-energy curves are plotted as a function of $V_{DD}$ and $V_{TH}$ for different stack depths, confirm this. A clear shift to lower threshold values for larger stack depths can be observed.

Slide 11.47
This observation is confirmed by comparing the energy-delay plots of a four-input NAND and an equivalent NAND–NOR implementation using only fan-in of 2 gates. Independent of the activity levels, the NAND4 implementation consumes less energy for the same performance.

Slide 11.48
Though this seems to be an encouraging line of thought, the overall impact is limited. Additional modifications in the gate topology are necessary for more dramatic improvements. One promising option to do so is offered by the pass-transistor logic (PTL) approach. Over the years, PTL has often been considered as a premier candidate for low-energy computation, mainly owing to its simple structure and small overhead. Even in the original papers on CPL (complementary pass-transistor logic) [Yano’90], it was suggested that the threshold voltages of the pass-transistors could be reduced to almost zero to improve performance with little impact on energy. At that time, standby power was not considered to be a major issue yet, and it turns out that sneak paths between inputs make this approach non-effective from a leakage perspective.

Yet, a basic observation stands: networks of switches – regardless of their complexity – on their own do not add any meaningful leakage, as they do not feature any direct resistive connections to either $V_{DD}$ or GND. The only leakage of such a network occurs through the parasitic diodes, and is
Controlling Leakage in PTL

Pass-Transistor Network

- Confine leakage to well-defined and controllable paths

- No leakage through the logic path
- No $V_{DD}$ and GND connections in the logic path
- Leverage complexity

[Ref: L. Alarcon, Joipe'07]

Sense-Amplifier Based Pass-Transistor Logic (SAPTL)

Pass-Transistor network

root node driver

S

outputs

S

Sense amplifier

timing control

Leakage path confined to root node driver and sense amplifier

Sense amplifier to recover delay and voltage swing

[Ref: L. Alarcon, Joipe'07]

small. The switches do nothing more (or less) than steering the currents between the input and output nodes.

Hence, a clever connection and control of the input and output sources could help to combine to attractive features: complex logic with very low thresholds and controllable on- and off-currents.

**Slide 11.49**

The sense amplifier based PTL (SAPTL) family represents an example of such a scheme [Alarcon'07]. In evaluation mode, current is injected into the root of a pass-transistor tree. The network routes the current to either the $S$ or the $\bar{S}$ node. A high-impedance sense amplifier, turned on only when the voltage difference between $S$ and $\bar{S}$ has built up sufficiently, is used to restore voltage levels and to speed up the computation. As a result, the gate can function correctly even at very low $I_{on}/I_{off}$ ratios. Observe that leakage current only occurs in the driver and sense amplifiers, and hence can be carefully controlled, independent of the logic complexity.
Slide 11.50
A more detailed transistor diagram is shown in this slide. The pass-transistor network is implemented as an inverse tree, and can be made programmable (just like an FPGA Look-up table or LUT) by adding an extra layer of switches. The sense amplifier output nodes are pre-charged high, and get conditionally pulled low during evaluation, effectively latching the result. Correct operation down to 300 mV has been demonstrated.

Slide 11.51
The energy–delay curves of equivalent functions in CMOS, transmission-gate logic, and SAPTL are plotted in this chart. The graph clearly demonstrates how each logic style has its own “sweet spot”. Most importantly though, it shows that pass-transistor based logic families extend the operational range of MOS logic to energy levels that are substantially below what can be accomplished with traditional complementary CMOS. This observation alone should be sufficient to motivate designers to explore the broad space of opportunities offered by alternative logic styles. We contend that the exploration space will expand even further with the emergence of new switching devices.

Slide 11.52
In summary, further scaling of the EOP is essential if further miniaturization of computing is to continue. There is definitely room for improvement, as indicated by the Von Neumann–Landauer–Shannon bound on digital switching. Operating the transistor in the weak- or moderate-inversion
Summary

- To continue scaling, a reduction in energy per operation is necessary
- This is complicated by the perceived lower limit on the supply voltage
- Design techniques such as circuits operating in weak or moderate inversion, combined with innovative logic styles, are essential if voltage scaling is to continue
- Ultimately the deterministic Boolean model of computation may have to be abandoned

The physical limits of scaling, it may be unwise to stick to this model – when all processes are statistical, maybe computation should be as well.

References

Books and Book Chapters

Articles

References (contd.)

Chapter 12
Low Power Design Methodologies and Flows

Slide 12.1
The goal of this chapter is to describe the methodologies and flows currently used in low-power design. It is one thing to understand a technique for achieving low power; it is another to understand how to efficiently and effectively implement that technique. Previous chapters have focused upon particular techniques and how they achieve energy efficiency. This chapter explores methodologies for implementing those techniques along with issues and trade-offs associated with those methodologies.

Slide 12.2
There is more to the story of low-power design than power minimization. A substantial amount of time and effort are needed to achieve that sought-after energy efficiency. One of the main challenges is that, in effect, the task of low-power design itself is a multi-variable optimization problem. As we will later see, optimizing for power in one mode of operation can actually cause increased power in other modes and, if one is not careful, the time and effort spent on various power-related issues can swell to match and even exceed that spent on basic functionality. Hence, the motivations for an effective low-power design methodology include minimizing time and effort in addition to minimizing power.

The first decade of power-efficient design was mostly devoted to the development of design technologies – that is, techniques to reduce power. With power efficiency becoming a prime design metric, at the same level of importance as area and speed, electronic design automation (EDA) companies are gradually getting engaged in the development of integrated design flows (and the supporting tools) for low power. This proves to be advantageous, as many design techniques, described in this book, are now becoming an integral part of the design flow, and, as such, are available to a broader audience. This is testified by the
Low Power Design Methodology – Motivations

- Minimize power
  - Reduce power in various modes of device operation
  - Dynamic power, leakage power, or total power

- Minimize time
  - Reduce power quickly
    - Complete the design in as little time as possible
  - Prevent downstream issues caused by LPD techniques
    - Avoid complicating timing and functional verification

- Minimize effort
  - Reduce power efficiently
    - Complete the design with as few resources as possible
  - Prevent downstream issues caused by LPD techniques
    - Avoid complicating timing and functional verification

Methodology Issues

- Power Characterization and Modeling
  - How to generate macro-model power data?
  - Model accuracy

- Power Analysis
  - When to analyze?
  - Which modes to analyze?
  - How to use the data?

- Power Reduction
  - Logical modes of operation
    - For which modes should power be reduced?
  - Dynamic power versus leakage power
  - Physical design implications
  - Functional and timing verification
  - Return on Investment
    - How much power is reduced for the extra effort? Extra logic? Extra area?

- Power Integrity
  - Peak instantaneous power
  - Electromigration
  - Impact on timing

---

**Slide 12.3**

There are a variety of issues to consider when developing a low-power design methodology.

The first issue pertains to **characterization and modeling**. In cell-based system-on-chip (SoC) design flows, each cell must have a power model in addition to the usual functional and timing models. Without a full set of sufficiently accurate models, the methodology’s effectiveness will be compromised.

The methodology must define the points in the design process at which **power analysis** will be performed; that is, is it only a major milestone check or something that is performed regularly and with great frequency, or something in between? The answer may depend upon the project’s goals and the complexity of the device being designed. Similarly, what will the results of the analysis be used for – is it simply to check whether the overall design is on target to meet a basic specification or to check against multiple operational modes, or to use as a guidance on how to reduce power further? Or perhaps, is the goal not power reduction at all, but instead verification of the power delivery network (also known as **power integrity**)? The answers to these questions will determine the frequency of and the level of detail needed from power analysis.

Similarly, **power reduction** efforts should be driven by specific project objectives. How many different power targets have been specified? What are the priorities of those targets relative to other project parameters such as die size, performance, and design time? The answers to these questions
help determine not only the particular power reduction techniques to be employed but also the methodologies used to implement them.

**Some Methodology Reflections**

- Generate required models to support chosen methodology
- Analyze power early and often
- Employ (only) as many LPD techniques as needed to reach the power spec
  - Some techniques are used at only one abstraction level; others are used at several
    - Clock Gating: multiple levels
    - Timing-stack redistribution: only physical level
- Methodology particulars dependent upon choice of techniques
  - Power gating versus Clock gating
  - Very different methodologies
- No free lunch
  - Most LPD techniques complicate the design flow
  - Methodology must avoid or mitigate the complications

**Power Characterization and Modeling**

- **Objective:** Build models to support low-power design methodology
  - Power consumption models
  - Current waveform models
  - Voltage-sensitive timing models
- **Issues**
  - Model formats, structures, and complexity
    - Example: Liberty-power
  - Run times
  - Accuracy

*Ref: Liberty*

**Slide 12.4**

Despite the various questions raised in contemplating the aforementioned issues, several truisms hold. Models are needed and can be generated during the earliest stages of the project, often well before usage is actually required. Power should be analyzed early and often so as to keep a close watch on it and to prevent surprises. The number of reduction techniques should be limited to only those that are needed to meet project power targets as each technique usually complicates other design tasks.

**Slide 12.5**

In cell-based SoC design flows, power models are needed for each cell used in the design. So an enabling task is to build the models required for the chosen methodology. For example, to calculate leakage power consumption, leakage currents must be characterized and modeled. To analyze voltage drop, basic average-power models can be used, but for best accuracy, current waveform models should be created for each cell.

Similarly, if the methodology calls for checking the effects of voltage drop on timing (more on this later), voltage-sensitive timing models need to be built.

Whichever models are built, they must be compatible with the tools to be used in the methodology. There are several standards, along with proprietary formats, for modeling power, and usually various modeling options exist for each format. The most widely used format for modeling power at the time of this writing is Liberty power. It provides for pin-based and path-based dynamic power modeling. In the former case, power is consumed whenever a particular pin on that cell transitions, whereas in the latter a full path – from an input transition to an output
transition – must be stimulated to consume power. In either case, power is represented as a single value, for a particular input transition time and output load pair, representing the total power consumed for the entire event. A recent extension to Liberty power, known as CCS (composite current source [Liberty Modeling Standard]) uses a time-varying current waveform to model power, instead of a single value.

Both formats support both state-dependent and state-independent leakage models. As the name implies, state-independent models use a single value to represent the leakage of the cell, independent of whatever state the cell might be in, whereas state dependent models contain a different leakage value for each state. The trade-off here is model complexity versus accuracy and evaluation time. A state-independent model sacrifices accuracy for fast evaluations and a compact model, whereas a state-dependent model provides the best accuracy. In most cases, for standard-cell type primitives, full state dependent models are preferred. State-independent models or limited state dependency models are often used for more complex cells or for those cells with more than about eight inputs.

![Power Characterization and Modeling](image)

Slide 12.6

A common flow for generating power models is shown here. This flow is almost identical to that for generating timing models. In effect, multiple SPICE, or SPICE-like, simulations are run on the transistor-level netlist for each cell primitive: one simulation for each input-to-output path/input transition time/output load combination, monitoring the current drawn from the supply. The resulting data, $I_P$, $I_{SC}$, and $I_{leakage}$ (in this case), are collected and formatted into a particular model structure. This flow is usually encapsulated into a characterization tool that automatically runs SPICE based upon user-supplied parameters such as the characterization conditions (PVT: process, voltage, temperature), data model table sizes, and types of characterization to perform (power, timing, noise, etc.). The simulation stimulus is automatically generated by exhaustively stimulating every potential input-to-output path. However, the ease of use of this approach is mitigated by its lack of scalability, as the number of simulations grows exponentially $O(2^n)$.

Note that this type of characterization can be performed on larger objects or functional blocks, such as memories, controllers, or even processors. In such applications, the characterization engine might be a high-capacity SPICE simulator, or a gate or RTL (register transfer level) power analysis tool. For objects that have a large number of states, a target model template is essential. Such a template specifies the specific logical conditions under which to characterize the target block. This utilizes the block designer’s knowledge to avoid the $2^n$ simulation runs.
A generalized low-power design flow involves both analysis and reduction at multiple points in the development cycle, although the precise motivations for each analysis and reduction activity will differ. In this abstracted view of a low-power design flow, the overall effort is divided into three phases: system-level design, RTL design, and implementation. The system-level design (SLD) phase is when most of the large design decisions are made, especially in regard to how particular algorithms will be implemented. As such, this is also the phase in which designers have the largest opportunities to influence power consumption.

The RTL design phase is when the decisions made during the system-level design phase are codified into executable RTL form. The RTL may be manually coded to match the system conceived in the earlier phase, or it may be directly synthesized from a system-level description in C, C++, or SystemVerilog.

The implementation phase includes both the synthesis of the RTL description into a netlist of logic gates and the physical implementation of that netlist. It also includes tasks associated with final signoff such as timing and power closure and power grid verification.

Slide 12.7

Power analysis has two different, albeit related, motivations. The first is to determine whether the power consumption characteristics meet the desired specification; the second is to identify opportunities, or to enable methods, for reducing the power if the specification is not met.

The technique for analyzing power is to simulate the design using a power simulator or create an estimate using an estimation tool.

The methodology is to do so regularly and automatically by creating a set of power regression tests as soon as possible in the overall development. Such a setup raises the awareness not just of the design’s overall power characteristics but also of the impact of individual design decisions. Having
the power data always available and regularly updated is a major help in accomplishing such analysis.

**Slide 12.9**

This “early and often” methodology raises some issues requiring consideration. A general trade-off exists between the amount of detail used to generate a power estimate and its accuracy—the earlier the estimate, the less detail is available and the less accuracy is possible. On the other hand, the later in the design process the design is analyzed, the more detailed the information (sized gates, extracted parasitics) available and the greater the accuracy. But this increased accuracy comes at the expense of longer run times and increasing difficulty in using the results to find power reduction opportunities, as the greater detail can obscure the big picture—a “can’t see the forest for the trees” situation. In addition, the later in the design process, the harder it is to introduce major changes with large power impact. A useful concept to keep in mind is the following: the granularity of the analysis should be on par with the impact of the design decisions that are being made.

**Slide 12.10**

A power analysis flow during the system-level design phase involves binding different parts of the system description to particular hardware elements. This slide shows a flow in which the binding is accomplished by synthesizing the ESL (electronic system level) description to RTL code. A simulation of the ESL description produces a set of activity data, often in the form of transaction traces. An RTL power estimator will read the RTL code and the transaction traces to calculate the power. Note that several other inputs are required as well: environmental data (such as power supply voltages and external load capacitances), technology data
(representing the target fabrication technology – this is often encapsulated in the power models for the cell primitives) and power models for any non-synthesizable IP blocks.

These “additional” inputs are also required to estimate power during other phases and at other abstraction levels as well.

It is worth observing that this picture presents an ideal scenario. Today, most SoC designs rarely use a fully-automated analysis flow in the early phases of the design process. Power analysis is often performed using spreadsheets, combining activity estimates for the different operational modes with design data obtained from previous designs, IP vendors, or published data.

**Slide 12.11**

This flow is similar to that shown for the *system phase*, but with several significant differences. First, the design is simulated and analyzed at a single abstraction level: RTL. Second, the power analysis uses activity data (nodal transition changes) instead of transaction data. These two differences result in longer run times for a given simulation period (or, equivalently, a shorter simulation period for the same run time) but provide better calculation accuracy. A third difference is that this flow shows several RTL simulations (resulting in several activity files and power reports); this represents the idea that modal operation will be explored during this phase.

**Slide 12.12**

This flow bears great resemblance to the flow shown for the *design phase*, the primary difference being that a *gate-level netlist* is used for analysis instead of the RTL design. Note that the simulation is still performed on the RTL design. Although it is certainly possible (and even simpler in terms of design automation technology) to obtain activities from a gate-level simulation, in practice the gate-level simulations are often difficult to
set up and too time-consuming to run. Using RTL simulation data, as shown here, enables power analysis over longer simulation periods with only a slight compromise in accuracy. Activity data for the nodes in the gate-level netlist that are not present in the RTL simulation are calculated probabilistically.

**Slide 12.13**

Shown here are the power analysis results for a microprocessor design, tracked over the course of the RTL design phase. In this particular case, power regressions were set up to run automatically every weekend, enabling the design team to easily see and understand where they were relative to the project goals. This type of attention is needed to cultivate an environment of power awareness, enabling the team – designers and managers alike – to efficiently pursue their power objectives. Two particular items should be noted. The power goal of this project was to achieve a 25% power reduction from the previous design; it can be seen that this goal was achieved by week 15. Second, during week 13, a change occurred that resulted in power actually increasing from the previous week’s results, promptly alerting the team that corrective action was in order. It can also be seen that the corrective action occurred by week 14.

**Slide 12.14**

For some projects, it is important only to know whether a particular power target is achieved or not. For others, the goal is to reduce power as much as possible. For the latter cases it is especially important to begin the process of low-power design during the system phase.

The primary low-power design objectives during the system phase are the minimization of \( f_{\text{eff}} \) (i.e., the effective switching frequency), which is the

---

**System-Phase Low-Power Design**

- **Primary objectives:** minimize \( f_{\text{eff}} \) and \( V_{\text{DD}} \)

- **Modes**
  - Modes enable power to track workload
  - Software programmable; set/controlled by OS
    - Hardware component needed to facilitate control
    - Software timers and protocols needed to determine when to change modes and how long to stay in a mode

- **Parallelism and Pipelining**
  - \( V_{\text{DD}} \) can be reduced, since equivalent throughput can be achieved with lower speeds

- **Challenges**
  - Evaluating different alternatives
product of the clock frequency and the average switching activity) and $V_{DD}$. These can be accomplished by various means, but the main idea is to explore different options and alternatives. One of the key techniques for the reduction of $f_{eff}$ is the use of modes, whereas $V_{DD}$ reduction can often be achieved using parallelism and/or pipelining to increase the raw throughput. In either case, the essential methodology challenge is the evaluation of the different alternatives in terms of their power consumption.

**Power-Down Modes – Example**

- Modes control clock frequency, $V_{DD}$, or both
  - Active mode: maximum power consumption
    - Full clock frequency at max $V_{DD}$
  - Doze mode: -10X power reduction from active mode
    - Core clock stopped
  - Nap mode: -50% power reduction from doze mode
    - $V_{DD}$ reduced, PLL & bus snooping stopped
  - Sleep mode: -10X power reduction from nap mode
    - All clocks stopped, core $V_{DD}$ shut off

- Issues and Trade-offs
  - Determining appropriate modes and appropriate controls
  - Trading off power reduction for wake-up time

[Ref: S. Gary, D&T'94]

the highest clock frequency, a technique known as *dynamic voltage and frequency scaling* (DVFS) (see earlier chapters). Multiple levels of power-down are possible, ranging from the four coarse levels shown here to an almost continuous range of levels possible with DVFS, with voltage steps as small as 20 mV.

However, controlling the power-down modes can be quite complicated involving software policies and protocols. In addition, there is often a trade-off between the amount of power reduction and the time needed to emerge from the power-down mode, known as the wake-up time. These and other trade-offs should be explored early, preferably in the system phase and no later than the design phase, to avoid late discoveries of critical parameters not meeting specifications. This need for early investigation is, of course, not unique to power, but it is perhaps more prevalent with power parameters as there is usually no single-large factor to address. Instead, the power consumption problem is the amalgam of millions of tiny power consumers.
Parallelism and Pipelining – Example

- Concept: maintain performance with reduced $V_{DD}$
  - Total area increases but each data path works less in each cycle
  - $V_{DD}$ can be reduced such that the work requires the full cycle time
  - Cycle time remains the same, but with reduced $V_{DD}$
- Pipelining a data path
  - Power can be reduced by 50% or more
  - Modest area overhead due to additional registers
- Paralleling a data path
  - Power can be reduced by 50% or more
  - Significant area overhead due to paralleled logic
- Multiple CPU cores
  - Enables multi-threaded performance gains with a constrained $V_{DD}$

- Issues and Trade-offs
  - Application: can it be paralleled or threaded?
  - Area: what is the area increase for the power reduction?
  - Latency: how much can be tolerated?

[Ref. A. Chandrakasan, JSSC’92]

---

Slide 12.16
Parallelism and pipelining can be employed to reduce power consumption, but at the expense of increased area (Chapter 5). Parallelism can be deployed at a relatively low level, such as paralleling a data path, or at a much higher level, such as using multiple processor cores in a single design, an example of which is the Intel Penryn™ multi-core processor. In either case, the delay increases resulting from the reduced supply voltages are mitigated by the enhanced throughput of the replicated logic.

---

Slide 12.17
The generalized low-power design flow during the system phase begins with the creation of the design, or at least a model of the design, in a high-level language such as C, C++, or SystemVerilog. The design is simulated under typical workloads to generate activity data, usually in the form of transactions, for use in power analysis. Several different versions of the model are created so as to evaluate and compare the power characteristics of each, thus enabling the designer to choose the lowest-power version or the best alternative. As an example of this flow, seven different versions of an 802.11a transmitter were synthesized from a SystemVerilog description at MIT [Davé'06]. Power ranged from 4 to 35 mW with a corresponding range in area of 5–1.5 mm².
Design-Phase Low-Power Design

- Primary objective: minimize $f_{\text{eff}}$
- Clock gating
  - Reduces / inhibits unnecessary clocking
    - Registers need not be clocked if data input hasn’t changed
- Data gating
  - Prevents nets from toggling when results won’t be used
  - Reduces wasted operations
- Memory system design
  - Reduces the activity internal to a memory
  - Cost (power) of each access is minimized

**Slide 12.18**
The key low-power design objective in the design phase is the minimization of $f_{\text{eff}}$. Here again this does not mean reducing the clock frequency, although it often involves a reduction in how often the clock toggles by implementing clock gating. Another technique of reducing $f_{\text{eff}}$ is the use of data gating. Similarly, a reduction in the number of accesses to a memory is an effective design-phase technique for reducing power.

**Slide 12.19**
Clock gating is the single-most popular technique for reducing dynamic power. It conserves power by reducing $f_{\text{eff}}$, which in turn reduces two different power components. The first is the power consumed in charging the load capacitance seen by the clock drivers, and the second is the power consumed by the switching of each register’s internal clock buffers. (virtually all common standard-cell registers have buffered clock inputs so as to produce the

true and complement clock signals needed by the basic latching structure.)

Two different flavors of clock gating are commonly used. Local clock gating involves gating individual registers, or banks of registers, whereas global clock gating is used to gate all the registers within a block of logic. Such a block can be relatively small, such as perhaps a few hundred instances, or it can be an entire functional unit consisting of millions of instances.

**Slide 12.20**
Several different methods are used to insert clock gating into designs. The easiest method is to use the logic synthesizer to insert the clock gating circuitry wherever it finds a logical structure involving a feedback multiplexer, as shown on the previous slide. This can work well for local gating, but is not applicable to global clock gating, as synthesizers generally cannot recognize or
Clock-Gating Insertion

- **Local clock gating: Three methods**
  - Logic synthesizer finds and implements local gating opportunities
  - RTL code explicitly specifies clock gating
  - Clock-gating cell explicitly instantiated in RTL

- **Global clock gating: Two methods**
  - RTL code explicitly specifies clock gating
  - Clock-gating cell explicitly instantiated in RTL

In the logical case, it is usually desirable to avoid such a simplistic approach for two reasons. The first is that clock gating does not always reduce power, as the additional logic inserted to gate the clock will consume extra power when the gated clock leaf node toggles. Thus, this may result in higher maximum sustained power — for instance, if the design has a mode in which few of the clocks are simultaneously disabled. Even the long-term time-averaged power can be higher with clock gating. If the enable is active for a high percentage of time, the power consumed by the extra logic may exceed the amount of power saved when the clock is disabled. The second reason to avoid “wherever-possible” clock gating is to prevent complicating the clock tree synthesis, which occurs later in the implementation phase. Excessive numbers (exceeding roughly a hundred) of gated clock leaves can make it very difficult to achieve low timing skew between the leaf nodes.

Clock Gating Verilog Code

- **Conventional RTL Code**

```verilog
always @ (posedge clk) begin  // form the flip-flop
    if (enable) q = din;
end
```

- **Low-Power Clock-Gated RTL Code**

```verilog
// only clock the register when enable is true
assign gclk = enable && clk;  // gate the clock
always @ (posedge gclk) begin  // form the flip-flop
    q = din;
end
```

- **Instantiated Clock-Gating Cell**

```verilog
// instantiate a clock-gating cell from the target library
clkpxl 11 .en(enable), .cp(clk), .gclk_out(gclk);
always @ (posedge gclk) begin  // form the flip-flop
    q = din;
end
```

The first snippet of Verilog RTL code shows a canonically enabled register that will become a register with a feedback multiplexer when synthesized without low-power optimizations, or a clock-gated register when synthesized using low-power synthesis features. However, low-power synthesizers usually clock gate all recognized opportunities, which is usually undesirable, for the reasons mentioned on the previous slide. One method of controlling which registers get synthesized with gated clocks is to explicitly define the gating in the RTL code, as shown in the middle snippet, and to turn automatic clock gating off during synthesis. Another method of explicitly defining the clock gating is to instantiate an integrated clock-gating cell from the target cell library as shown in the third code snippet.
Alternately, a synthesizer can be provided with constraints explicitly specifying which registers should be clock gated and which should not.

**Slide 12.22**
Most implementations of clock gating employ some version of the logic shown here to prevent glitches on the gated clock output. The RTL code, when synthesized, will produce a gated clock with a glitch prevention latch as shown. This logic is often implemented together in a single library cell known as an integrated clock gating cell.

**Slide 12.23**
Data gating is another method of reducing $f_{\text{eff}}$. Whereas clock gating reduces $f_{\text{eff}}$ for clock signals, data gating focuses upon non-clock signals. The general concept is to prevent signal toggles from propagating through downstream logic if those toggles result in unused computations. The particular example shown here is known as operator isolation; the multiplication operator is isolated so that it will not be called upon to operate unless its results will be selected to pass through the downstream multiplexer. Power is saved by preventing unused operations.
Data-Gating Insertion

- Two insertion methods
  - Logic synthesizer finds and implements data-gating opportunities
  - RTL code explicitly specifies data gating
    - Some opportunities cannot be found by synthesizers

- Issues
  - Extra logic in data path slows timing
  - Additional area due to gating cells

area for the gating cells, and the gating cells introduce additional delays into the timing path, which may be undesirable depending upon timing requirements. For these reasons, it is sometimes desirable to explicitly embed the data gating into the RTL code.

Data-Gating Verilog Code: Operand Isolation

- Conventional Code
  
  ```verilog
  assign muxout = sel ? A : A*B ; // build mux
  ```

- Low-Power Code
  
  ```verilog
  assign multmA = sel & A ; // build and gate
  assign multmB = sel & B ; // build and gate
  assign muxout = sel ? A : multmA*multmB ;
  ```

multiplier are latched – keeping the old data intact and thus avoiding unnecessary activity.

Slide 12.25
Shown here is an example of conventional RTL code of a data-gating opportunity. An operator isolation capable synthesizer recognizes this gating opportunity and automatically inserts the isolation logic. The low-power code snippet shows an example of RTL code that describes the data gating explicitly, obviating the need for an operator isolation capable synthesizer. Observe that activity would be reduced further if the inputs of the

Slide 12.26
Memory systems often represent significant opportunities for power reduction, as they usually consume considerable portions of a design’s overall power budget. It is often possible to minimize this power by implementing some form of memory banking or splitting. This technique involves splitting the memory into several sections, or banks, so as to minimize the extent of the memory that must be activated for a particular access, thus reducing the power consumed for any particular access.
Another technique for lowering memory system power is to reduce the number of accesses through such techniques as storing intermediate results in a local register instead of writing to memory or restructuring algorithms such that fewer memory accesses are needed, although this latter technique is best addressed during the system phase.

memory, we now have two accesses to a $16\,\text{K} \times 32$ memory, performed in parallel, with a later multiplexer toggle.

Note that this particular banking strategy saves power only for an application that sequentially accesses memory. For different access patterns, different banking techniques are needed. In terms of methodology, the key issue here is to analyze the access patterns; the knowledge gained from that effort will illuminate opportunities to restructure the memory system to reduce its power consumption.

Multiple techniques exist for reducing power during the implementation phase; however, they are generally limited in terms of how much total power can be saved as so much of the design is fixed by...
Implementation Phase Low-Power Design

Primary objective: minimize power consumed by individual instances
- Low-power synthesis
  - Dynamic power reduction via local clock gating insertion, pin-swapping
- Slack redistribution
  - Reduces dynamic and/or leakage power
- Power gating
  - Largest reductions in leakage power
- Multiple supply voltages
  - The implementation of earlier choices
- Power integrity design
  - Ensures adequate and reliable power delivery to logic

easily result in many Amperes coursing through the power grid. One must be careful to verify that the chip will function correctly with such large currents, and rapid changes in those currents, occurring.

Smack Redistribution

- Objective
  - Reduce dynamic power or leakage power
  - or both by trading off positive timing slack
  - Physical-level optimization
    - Best optimized post-route
    - Must be noise-aware
- Dynamic power reduction by cell resizing
  - Cells along non-speed critical path resized
    - Usually downsized, sometimes upsized
    - Power reduction of 10–15%
- Leakage power reduction by $V_{TH}$ assignment
  - Cells along non-speed critical path set to High
  - Leakage reduction of 20–60%
- Dynamic & leakage power can be optimized independently or together

[Ref: Q. Wang, TCAD’02]

this time. Nevertheless, efforts at this level are important in an overall low-power design flow, especially for leakage reduction.

Note that this phase also includes tasks focused on power integrity. Although, strictly speaking, these efforts are not focused on power reduction, they are nonetheless essential components in an overall methodology, as the use of very low supply voltages can

Slide 12.29
Slack redistribution is a particularly common, and conceptually straightforward, implementation phase power reduction technique, involving the trade-off between positive timing-sack and power. As synthesizers generally work to speed up all timing paths, not just the critical paths, many instances on the non-critical paths end up being faster than they need to be. Slack redistribution works by slowing down the instances off the critical path, but only to the extent that the changes do not produce new timing violations (see Chapters 4 and 7). The changes produce a new slack timing histogram, as shown in the chart, hence the moniker. Typical changes reduce the drive strength of a particular instance (usually saving dynamic power) or increase the instance’s threshold voltage (reducing leakage power).

Slide 12.30
An example of slack redistribution, as applied to dynamic power reduction, is shown here. This optimization can be performed either before or after routing, but is usually performed after routing as actual extracted parasitics can be used for delay calculation to provide the most accurate timing analysis and hence the best optimization results. Note that after this optimization, the design must
Dynamic Power Optimization: Cell Resizing

- Positive-Slack Trade-Off for Reduced Dynamic Power
  - Objective: reduce dynamic power where speed is not needed
  - Optimization performed post-route for optimum results
  - Cells along paths with positive slack replaced with lower-drive cells
    - Switching currents, input capacitances, and area are all reduced
    - Incremental re-route required – new cells may have footprints different from the previous cells

Leakage Power Optimization: Multi-$V_{TH}$

- Trade Off Positive Slack for Reduced Leakage Power
  - Objective: reduce leakage power where speed is not needed
  - Optimization performed post-route for optimum results
  - Cells along paths with positive slack replaced with High-$V_{TH}$ cells
    - Leakage currents reduced where timing margins permit
    - Re-routing not required – new cells have same footprint as previous cells

Slide 12.31
Another example of slack redistribution targets leakage power reduction. Also best-performed after routing, this optimization replaces instances off the critical path with same-sized instances but with higher threshold voltages, thus reducing the leakage power with each replacement.

It should be noted that one side effect of replacing a low-$V_{TH}$ cell with a high-$V_{TH}$ cell (or replacing a high-drive cell with a low-drive version) is that the net driven by the replaced cell becomes more susceptible to signal integrity (SI) noise effects, such as coupling delays and glitches. Thus, it is essential to verify noise immunity after each slack redistribution optimization.

Slide 12.32
Two slightly different slack redistribution flows are commonly employed. The left-hand flow illustrates a sequential flow, in that timing, noise, and power are each verified in sequence. This flow works, but tends to require multiple iterations through the entire flow as noise fixes and power optimizations tend to “fight” each other – changes made by the noise fixer tend to be reversed by the power optimizer, and vice versa. For example, a common noise fix is to increase the drive strength of a particular instance so that it is less sensitive to coupling delays. However, when the power optimizer sees this, it may conclude that particular instance to be oversized and either downsizes it or raises its threshold voltage leading to a potential flow convergence issue.
Slack Redistribution Flows

The right-hand flow avoids this issue through the use of concurrent optimizing tools, software that concurrently checks multiple parameters during each optimization. In this case, the noise optimizer is timing-aware – any changes it makes to fix noise problems will not break timing – and the power optimizer is both noise- and timing-aware – any changes it makes to reduce power will not break timing or introduce any new noise problems. These tools tend to be more complex than those employed in the sequential flow, but result in faster timing/noise/power closure as the number of iterations through the flow is greatly reduced.

Slack Redistribution: Trade-Offs and Issues

- **Yield**
  - Slack redistribution effectively turns non-critical paths into critical or semi-critical paths
  - Increased sensitivity to process variation and speed faults
- **Libraries**
  - Cell resizing needs a fine granularity of drive strengths for best optimization results → more cells in the library
  - Multi-$V_{TH}$ requires an additional library for each additional $V_{TH}$
- **Iterative loops**
  - Timing and noise must be re-verified after each optimization
  - Both optimizations increase noise and glitch sensitivities
- **Done late in the design process**
  - Difficult to predict in advance how much power will be saved
  - Very much dependent upon design characteristics

Slide 12.33

Slack redistribution flows are very popular as they are fully automatic, but they introduce some issues of their own. Perhaps the most significant of these is that it is difficult to predict in advance just how much power will be saved, as the results are so design-dependent. In addition, the resulting slack distribution is narrower with more paths becoming critical, which makes the design more vulnerable to the impact of process variations. For these reasons, slack redistribution is rarely employed as the only low-power design method. Instead, it is usually used in addition to several other techniques and methods employed earlier in the design process.
**Power Gating**

- **Objective**
  - Reduce leakage currents by inserting a switch transistor (usually high-$V_{th}$) into the logic stack (usually low-$V_{th}$)
  - Switch transistors change the bias points ($V_{dd}$) of the logic transistors
- **Most effective for systems with standby operational modes**
  - 1 to 3 orders of magnitude leakage reduction possible
  - But switches add many complications

**Power Gating: Physical Design**

- **Switch placement**
  - In each cell?
    - Very large area overhead, but placement and routing is easy
  - Grid of switches?
    - Area-efficient, but a third global rail must be routed
  - Ring of switches?
    - Useful for hard layout blocks, but area overhead can be significant

---

**Slide 12.34**

Power gating is perhaps the most effective technique for reducing leakage power, resulting in savings ranging from 1 to 3 orders of magnitude (see Chapter 7). Conceptually very simple, power gating adds many complications to the design process. It should also be noted that power gating is not appropriate for all applications. To be effective, power gating should only be used on those designs that contain blocks of logic that are inactive for significant portions of time.

---

**Slide 12.35**

One of the major decisions to be made regarding power-gating implementation concerns switch placement. Shown here are three different switch placement styles: switch-in-cell, grid-of-switches, and ring-of-switches.

The switch-in-cell implementation uses a switch transistor in each standard cell. In practice, the standard cells are designed up front, each containing a switch, so that each individual instance is power-gated. This is sometimes referred to as fine-grained power gating. This has the advantage of greatly simplifying physical design, but the area overhead is substantial, almost equaling that of the non-power-gated logic area.

The grid-of-switches implementation style involves placing the switches in an array across the power-gated block. This generally results in three rails being routed through the logic block: power, ground, and the virtual rail. For this reason, this is often the desired style when state must be retained in registers within the block, as state retention registers need access to the real rails as well as the virtual rails.
The ring-of-switches implementation style places, as the name implies, a ring of switches around the power-gating block; the switches “break” the connection between the external real rail and the internal virtual rail. This style is often utilized for legacy designs for which the original physical design should not be disturbed.

**Power Gating: Switch Sizing**

- Trade-off between area, performance, leakage
  - Larger switches → less voltage drop, larger leakage, more area
  - Smaller switches → larger voltage drop, less leakage, less area

![Graph showing trade-off between area, performance, leakage](Ref: J. Frenkel, Springer/07)

**Slide 12.36**

Perhaps the biggest challenge in designing power-gated circuits is **switch sizing**, owing to the relationship between switch size, leakage reduction, and performance and reliability impact. On the one hand, smaller switches are desirable as they occupy less area. However, smaller switches result in larger switch resistance, which in turn produces a larger voltage drop across them. This larger voltage drop is undesirable as it results in increased switching times and degraded signal integrity. On the other hand, though larger switches impact performance less, they occupy more area. In addition, leakage reduction is smaller with larger switches, as the smaller voltage drop reduces the body effect on the logic transistors. This relationship between virtual-rail voltage drop (a proxy for switch size), delay degradation, and leakage reduction is clearly shown in the left-hand chart, while the relationship between switch area and voltage drop is shown on the right.

**Power Gating: Additional Issues**

- Library design: special cells are needed
  - Switches, isolation cells, state retention flip-flops (SRFFs)
- Headers or Footers?
  - Headers better for gate leakage reduction, but ~2X larger
- Which modules, and how many, to be power gated?
  - Sleep control signal must be available, or must be created
- State retention: which registers must retain state?
  - Large area overhead for using SRFFs
- Floating signal prevention
  - Power-gate outputs that drive always-on blocks must not float
- Rush currents and wake-up time
  - Rush currents must settle quickly and not disrupt circuit operation
- Delay effects and timing verification
  - Switches affect source voltages which affect delays
- Power-up & power-down sequencing
  - Controller must be designed and sequencing verified

(verify power-up and power-down sequencing, verifying voltage drop and wake-up time limits).

**Slide 12.37**

Numerous additional issues must also be addressed in the design of power-gated circuitry. Some of these are planning issues (the choice of headers or footers, which registers must retain state, the mechanism by which state will be retained), some others are true implementation issues (inserting isolation cells to prevent the power-gated logic outputs from floating), and the rest are verification-oriented...
A generalized flow for implementing power gating is shown here. Note that the choice of state retention mechanism dictates, to a certain extent, the design of the cell library – if the desired mechanism is to use *state retention flip-flops* (SRFFs), then those primitives must be present in the library. Further, this choice can influence whether the ring-of-switches or grid-of-switches floorplan is chosen. Alternatively, the choice of floorplan can force the issue. For example, the use of a ring-of-switches floorplan makes it awkward to use SRFFs, and a scan-out (for retention) and scan-in (for state restoration) approach may be preferable. In any event, the key concept is that many of these issues are intimately inter-related, thus requiring careful consideration up-front prior to proceeding in earnest on physical implementation.

**Multi-$V_{DD}$**

- **Objective**
  - Reduce dynamic power by reducing the $V_{DD}^2$ term
    - Higher supply voltage used for speed-critical logic
    - Lower supply voltage used for non-speed-critical logic

- **Example**
  - Memory $V_{DD} = 1.2$ V
  - Logic $V_{DD} = 1.0$ V
  - Logic dynamic power savings = 30%

Whereas power gating addresses leakage power reduction, multi-$V_{DD}$ (or multiple supply voltages – MSV) addresses dynamic power reduction. Shown here is a plain vanilla example of a design with two voltage domains, but several variants are possible. One variant simply uses more – three or four – voltage domains. Another variant, known as dynamic voltage scaling (DVS) uses time-varying voltages such that the supply is kept at a high value when maximum performance is needed, but reduces the supply voltage at other times. A more complex variant, dynamic voltage and frequency scaling (DVFS) adjusts the clock frequency along with the supply voltage. As one might imagine, while reducing power, these techniques increase the overall complexity of design and verification.
Multi-\(V_{DD}\) Issues

- Partitioning
  - Which blocks and modules should use which voltages?
  - Physical and logical hierarchies should match as much as possible
- Voltages
  - Voltages should be as low as possible to minimize \(CV_{DD}\)
  - Voltages must be high enough to meet timing specs
- Level shifters
  - Needed (generally) to buffer signals crossing islands
    - May be omitted if voltage differences are small, - 100 mV
  - Added delays must be considered
- Physical design
  - Multiple \(V_{DD}\) rails must be considered during floorplanning
- Timing verification
  - Sign-off timing verification must be performed for all corner cases across voltage islands
    - For example, for two voltage islands \(V_i, V_o\)
    - Number of timing verification corners doubles

Slide 12.40
Similar to power gating, the use of multiple supply voltages poses a variety of issues, which can be categorized as planning-, implementation-, and verification-oriented. While most of the complications occur during the implementation phase, the choice of which particular voltages to use, and on which blocks, should be made during the system phase. During implementation, the design will be floorplanned, placed, and routed taking into consideration the multitude of unique supplies (ground is usually shared amongst the different power domains). Level shifters must be inserted to translate voltage levels between domains and, depending on the level shifter design, may have placement constraints regarding which domain – driving or receiving – they can be placed in.

Verification becomes much more complicated, as timing analysis must be performed for all PVT corner cases involving signals crossing across power islands.

Multi-\(V_{DD}\) Flow

- Determine which blocks run at which \(V_{DD}\)
- Multi-voltage synthesis
- Determine floorplan
- Multi-voltage placement
- Clock tree synthesis
- Route
- Verify timing

Slide 12.41
A generalized flow for implementing a multi-\(V_{DD}\) design is shown here. The overall flow is very similar to a standard design flow, but several tasks deserve particular attention. First, and perhaps foremost, planning is needed both in terms of logically determining which blocks will run at which voltage and in terms of how the power rails will be routed and the physical blocks will be placed. Multi-voltage synthesis will logically insert level shifters, as appropriate, and the placement of those level shifters must be considered during the subsequent multi-voltage placement. Clock tree synthesis must be multi-voltage aware; that is, it must understand that a clock buffer placed in one voltage domain will have different timing characteristics than the same buffer placed in a different voltage domain, and it should use that knowledge for managing latency and skew. Finally, after routing, timing must be verified using timing models characterized at the particular operating voltages for each different power domain.
Power Integrity Methodologies

- **Motivation**
  - Ensure that the power delivery network will not adversely affect the intended performance of the IC
    - Functional operation
    - Performance – speed and power
    - Reliability

- **Method**
  - Analyze specific voltage drop parameters
    - Effective grid resistances
    - Static voltage drop
    - Dynamic voltage drop
    - Electromigration
  - Analyze impact of voltage drop upon timing and noise

---

**Slide 12.42**

Power integrity is a concern in all integrated circuits, but it is an especially heightened concern in power-optimized devices, as many of the reduction techniques tend to stress the power delivery network and reduce the noise margins of the logic and memory. For example, power-gating switches insert additional resistance in the supply networks, making them particularly susceptible to excessive voltage drop. Clock gating, with its cycle-by-cycle enabling, often leads to large current differentials from one cycle to the next. Even basic operation at low supply voltages results in huge on-chip currents. Consider a 3 GHz Intel Xeon™ processor, consuming 130 W while running on a 1.2 V power supply – the on-chip supply currents exceed 100 A!

---

**Slide 12.43**

The basic idea behind power integrity verification is to analyze the equation $V(t) = I(t) * R + C * \frac{dv}{dt} * R + L * \frac{di}{dt}$, and to determine its effects upon the overall operation of the circuit.

A power integrity verification flow consists of several critical steps, employed in a successive-refinement approach. First, the grid is evaluated for basic structural integrity by checking the resistance from each instance to its power source (effective resistance check). Second, static currents (also known as average, or effective dc, currents) are introduced into the grid to check for basic voltage drop (static voltage drop analysis). Next, time-dependent behavior of the grid is analyzed (dynamic voltage drop analysis). Finally, the effects of voltage drop upon the behavior of the circuit are verified (voltage-aware timing and SI analysis). Along the way, if any of the analyses indicate problems, repairs or optimizations can be made. Prior to routing, the repairs are usually implemented by several mechanisms such as strap resizing, instance movement to spread peak currents, and decoupling-capacitor insertion. After routing, the latter two techniques are used most frequently.
Power Integrity: Effective Resistance Check

- **Motivation**
  - Verify connectivity of all circuit elements to the power grid
  - Are all elements connected?
  - Are all elements connected to the grid with a low resistance?

- **Method**
  - Extract power grid to obtain $R$
  - Isolate and analyze $R$ in the equation $V(t) = R_i + C \cdot \frac{dV}{dt} + R + L \cdot \frac{dI}{dt}$

Slide 12.44
The first step in the flow is to verify the connectivity of each instance to the grid by computing its effective resistance to the power source. In effect, this isolates the $R$ term in the voltage drop equation so that it can be checked, individually, for every instance. The result of an effective resistance analysis is a histogram indicating the number of instances with a particular effective resistance. Note the outliers in this chart, as they indicate a much higher resistance than all the others, and hence highlight problems such as missing vias or especially narrow rail connections. A well formed, errorfree power delivery network will produce a resistance histogram with a well-behaved distribution without any outliers.

This analysis, run without the need for any stimuli or current calculations, is computationally fast and, as it covers all instances, can easily highlight grid connectivity issues that are difficult to find by other methods.

Slide 12.45
After verifying grid connectivity, the next step is to analyze voltage drop. However, to do so the circuit must be stimulated, and, as power is a strong function of activity, a high-activity cycle should be chosen for analysis. This requires a careful selection from a simulation trace as indicated here.

Vectorless stimulation can be an attractive alternative to simulation-based stimuli as it obviates the need for lengthy simulations and vector selection. However vectorless analysis presents its own set of issues, such as choosing the vectorless set-up conditions like total power consumption targets or activity percentages. Also, the analysis may be overly-pessimistic, resulting in over-sizing (and potentially an increase in leakage power).
Effective-resistance analysis, as the former does not check the connections of every instance (because realistic stimuli do not activate all instances).

A useful metric to consider is the effective voltage, or the difference between $V_{DD}$ and $V_{SS}$, seen by individual instances. The effective voltage, considering static voltage drop on both the $V_{DD}$ and $V_{SS}$ rails, should not deviate from the ideal values by more than 2–3%.

Dynamic voltage drop analysis full RLC models should be used. Instead of a single solve of the $V(t)$ equation, multiple solves are performed, one for each specified time step, much like SPICE, to compute the time-varying voltage waveforms for all power grid nodes. The four plots on the bottom are from successive time steps of the same dynamic voltage drop computation – the progression illustrates typical voltage drop variations over time.

A maximum effective dynamic voltage drop of 10% is often considered acceptable, although less is obviously better owing to the deleterious effects of voltage drop upon delays.
the parasitic netlist, consisting of the RC rail models and an RLC package model.

Decaps are usually formed by either of two structures: a metal-to-metal capacitor or a thin gate-oxide capacitor. The latter is the prevalent construction as the capacitance per unit area is much higher than that for metal-to-metal capacitors. However, beginning at 90 nm and becoming substantially worse at 65 nm, gate leakage through the thin gate-oxide capacitors has become a concern. This has prompted the avoidance of decap “fill”, which filled all otherwise-unused silicon area with decoupling capacitors, and instead has been a motivation for optimization of the number of decaps inserted and where they are placed.

Slide 12.49
A second motivation for decap optimization is that decap effectiveness is a strong function of proximity to the aggressor. For a decoupling capacitor to prevent a transient current-demand event from disturbing the power grid, the decap’s response time must be faster than that of the event itself. That is, the resistance through which the current will flow from the decap to the aggressor must be sufficiently low so as to provide the charge at the demanded di/dt rate. From a physical perspective, the decap must be close to the aggressor. This can be seen in the chart which compares the results of two voltage drop analyses, one before placement optimization and one after. The post-optimization results used fewer decaps than the pre-optimization results, yet effectively
improved the worst voltage drop. The use of fewer decaps means that the leakage current due to all the decaps will also be reduced, assuming the use of thin gate-oxide decaps.

Decoupling capacitors can be inserted before or after routing; however, in either case a voltage drop analysis is needed to determine the location and decoupling capacitance requirements of the voltage drop aggressors.

**Slide 12.50**
The final step, after ensuring that all power parameters have been met for all modes, is to verify timing. This verification step is, of course, independent of whether low-power design techniques have been employed or not. However, owing to the low voltages and the impact upon timing of even small supply voltage deviations, the validation of the impact of voltage drop is especially important.

Two different methods are used to check the impact of voltage drop. The first, and most commonly used, is to check whether the worst-case voltage drop is within the library timing characterization conditions; that is, the actual voltage drop value is of little concern as long as it is within the voltage drop budget assumed during the up-front timing characterization.

The second method is to run static timing analysis using delays computed with actual voltages resulting from a dynamic voltage drop analysis. Such a voltage-sensitive timing analysis can illuminate timing problems that may have otherwise been uncovered, as illustrated in this chart.

**Slide 12.51**
Low-power design is more than running a power optimization tool. It especially requires designers’ creativity in crafting power-efficient algorithms, architectures, and implementations. As the design progresses through the various design phases, increasing levels of power-aware design automation can aid the pursuit of particular power goals. The effectiveness of such automation is dependent upon the chosen methodologies, but several truisms pretty much hold for all SoC design projects. Library cells must be characterized for power. Power should be analyzed early and often. Power consumption can be reduced at all abstraction levels and during all phases, but the best opportunities for major reductions occur during the earliest design phases. Finally, voltage drop concerns must be addressed before tape-out, particularly with regard to effects upon timing and noise characteristics.

As mentioned in the beginning of the chapter, the emergence of power efficiency as a primary design concern has caused the electronic design automation industry to increasingly focus on design flows and tools incorporating power as a primary criterion. Many of the outcomes have been described in this chapter. Yet, it is clear that the effort still falls short
**Summary – Low Power Methodology Review**

- Characterization and modeling for power
  - Required for SoC cell-based design flows

- Power analysis
  - Run early and often, during all design phases

- Power reduction
  - Multiple techniques and opportunities during all phases
  - Most effective opportunities occur during the early design phases

- Power integrity
  - Voltage drop analysis is a critical verification step
  - Consider the impact of voltage drop upon timing and noise

---

### Some Useful References

**Books and Book Chapters**

**Articles and Web Sites**
Chapter 13
Summary and Perspectives

Summary and Perspectives
Jan M. Rabaey

Slide 13.1
In this book, we have been exploring a broad range of technologies to address the power and energy challenges in digital integrated circuit design. Many of these techniques have only been developed over the past decade or so. Yet, one cannot help wondering where the future may lead us. In this last chapter, we present some short summaries of the developments in low-power design over the past years, where the state-of-the-art is today, and what may be in store for tomorrow.

Low-Power Design Rules – Anno 1997

- Minimize waste (or reduce switching capacitance)
  - Match computation and architecture
  - Preserve locality inherent in algorithm
  - Exploit signal statistics
  - Energy (performance) on demand
- Voltage as a design variable
  - Match voltage and frequency to required performance

More easily accomplished in application-specific than in programmable devices
[Ref. J. Rabaey, Inelf'97]

Slide 13.2
It is fair to state that the main developments in power reduction in the early 1990s can be classified under two headers: cutting the fat, and turning the supply voltage into a design variable.

Indeed, before power became an issue, energy waste was rampant (Does this not seem to bear an eerie resemblance to the world at large today?). Circuits were oversized, idle modules were still clocked, and architecture design was totally driven by performance. The supply voltage and its distribution...
grid were considered sacred and untouchable. And the idea that general-purpose computing was “somewhat” (three orders of magnitude!) inefficient was a shocker.

Both of the above-mentioned concepts are now main-stream. Most designs are doing away with any excess energy consumption quite effectively, and multiple and, sometimes, variable supply voltages are broadly accepted.

Slide 13.3
The emergence of leakage as a substantial source of power dissipation came as somewhat of a surprise in the late 1990s, more specifically with the introduction of the 130 and 90 nm technology nodes. Few roadmaps had foreseen this. Solutions were put forward and adopted swiftly.

The most important one was the adoption of power domains. They are essentially an extension of the voltage domains that were introduced earlier, and, as such, made the concept of dynamic supply and body voltage management a reality in most SoC and general-purpose designs. In a sense, this is the root of the idea of runtime optimization, a technique that is now becoming one of the most prevalent tools to further improve energy efficiency.

Designers also learned to live with leakage, and to use it to their advantage. For instance, when a circuit is active, allowing a healthy dose of leakage is actually advantageous, as was predicted by Kirsten Svensson and his group in Linkoping in 1993 [Liu’93]. Finally, the realization that memories are the prime consumers of standby power (which is a major issue in mobile devices) pushed the issue of energy-efficient memory design to the forefront.

Slide 13.4
With all the low-hanging fruit picked, it became clear that further improvements in energy efficiency could only be accomplished through novel and often-disruptive design solutions.

Although the concept of using concurrency to improve energy efficiency (or to improve performance within a constant power budget) was put forward in the early 1990s [Chandrakasan92], it was not until the beginning of the 2000s that the idea was fully adopted in the general-purpose computing world – it most often takes a major disaster for disruptive ideas to find inroads. Once the dam bursts, there is no holding back however. Expect large (and even huge) amounts of concurrency to be the dominating theme in the architectural design community for the foreseeable future.

A second contemporary evolution is that the concept of runtime optimization is now coming to full fruition. If energy efficiency is the goal, it is essential that circuits and systems are always functioning at the optimal point in the energy-delay space, taking into account the varying environmental conditions, activity level, and design variations. In an extreme form, this means that the traditional “worst-case” design strategy is no longer appropriate. Selecting the operating
condition of a circuit for the worst case comes with major overhead. Allowing occasional failures to happen is ok, if the system can recuperate from them.

It also seems obvious that we continuously need to explore ways of further scaling the supply voltage, the most effective knob in reducing energy consumption. Ultra low voltage design will continue to be a compelling topic over the next few decades.

Finally, we should realize that further reduction in energy per operation will ultimately challenge the way we have been doing digital computation for the past six or seven decades. Based on the Boolean–von Neumann–Turing principles, we require our digital engines to execute computational models that are entirely deterministic. When signal-to-noise ratios get extremely small and variances large, it is worth exploring the opportunities offered by statistical computational models. Though these may not fit every possible application domain, there are plenty of cases where it is very effective, as is amply demonstrated in nature.

The latter observation is worth some further consideration. Design tools and technologies have always been constructed such that the design statistics were hidden or could be ignored. However, of late this is less and less the case. Statistical design is definitely becoming of essence, but, unfortunately, our design tools are ill-equipped to deal with it. Analyzing, modeling, abstracting, composing, and synthesizing distributions should be at the core of any low-power design environment.

### Slide 13.5
In the remainder of this “perspectives” chapter, we briefly discuss some concepts worth watching, none of which were discussed so far.

### Slide 13.6
From an energy perspective, the ideal switching device would be a transistor or a switch that has an infinite sub-threshold slope, and a fully deterministic threshold voltage. Under such conditions, we
Novel Switching Devices

- Nanotechnology brings promise of broad range of novel devices
  - Carbon-nanotube transistors, NEMS, spintronics, molecular, quantum, etc.
  - Potential is there – long-term impact unclear
  - Will most probably need revisiting of logic design technology
- Outside-the-box thinking essential

could continue to scale supply and threshold voltages, maintaining performance while eliminating static power consumption. Trying to accomplish this with semiconductor switches is most probably a futile exercise. In Chapter 2, we introduced a couple of device structures that may be able to reduce the sub-threshold slope below 60 mV/decade.

Hence, it is worth considering whether some of the device structures that have been explored within the realm of nanotechnology may offer relief. Researchers have been proposing and examining a wide array of novel devices operating on concepts that are vastly different from that of the semiconductor transistors of today. Although this indeed has shown some interesting potential (e.g., spintronics operating on magnetism rather than electrostatics), the long-term prospective of most of these devices with respect to building digital computational engines remains unclear. The next digital switch may very well emerge from a totally unexpected side.

Example: Nano-Mechanical Relays

- Minimum energy in CMOS limited by leakage
  - Even if it had a perfect (zero leakage) power gate
- How about a nano-scale mechanical switch?
  - Infinite $R_{on}$, low $R_{off}$

[Ref: H. Kam, UCB’08]

Slide 13.7

This is illustrated by the following example. Recently, a number of researchers have been exploring the idea of using a mechanical switch. This indeed seems to be a throwback to the beginning of the 20th century, when relays were the preferred way of implementing digital logic. Technology scaling has not only benefited transistors, but has also made it possible to reliably manufacture a broad range of micro- and nano-electromechanical systems (MEMS and NEMS, respectively) of increasing complexity. A micro-mechanical switch, which reliably opens and closes trillions of times, would bring us closer to the ideal switch (at least for a while ...). The main advantage of the mechanical switch is that its on-resistance is really small (on the order of 1 Ω) compared to that of a CMOS transistor (which is on the order of 10 kΩ for a minimum-size device), whereas the off-current also is very small (air is the best dielectric available).

The challenge, however, is to produce a device that can reliably switch trillions of times. At the nanoscale level, atomic forces start to play, and stiction can cause a device to be stuck in the closed
position. Successful industrial MEMS products, such as the Texas Instruments’ DLP displays, have demonstrated that these concerns can be overcome.

Slide 13.8
The availability of a device with very low on-resistance has a profound impact on how logic can and should be built. Instead of the shallow-logic structures of today, complex deep logic becomes a lot more attractive. The (in)famous “fan-out of four” rule that governs logic design today no longer holds. But most importantly, it may help to move the energy–delay bounds into new territory. This is illustrated in this slide, where the energy–delay optimal curves of 32-bit adders, implemented in 90 nm CMOS and 90 nm relay logic, are compared. The concept of relay logic has been around for ages – however, until now, it was confined to the domain of bulky electromagnetic relays. With the advent of MEMS switches, it may rise to the foreground again. And, as is evident from the chart, it has better potential of getting us closer to the minimum energy bound than any other device we have seen so far. Only one “minor” detail: we have to figure out how to make it work in a scalable and reliable way.

This example shows only one possible way of how novel devices may disruptively change the low-energy design scene. We are sure that other exciting options will emerge from the large cauldron of nanotechnology in the coming decades.

Slide 13.9
One recurring topic in the low-energy research community is the search for more efficient ways of charging and discharging a capacitor. In fact, if we could do this without any heat dissipation, we may even be capable of designing a logic that consumes virtually no energy (!!!). A clue on how this might be done was given in Chapter 3, when we discussed adiabatic charging: If a capacitor is charged extremely slowly, virtually no energy is dissipated in the switch (see also Chapter 6). This idea created a lot of excitement in the 1990s and led to a flurry of papers on adiabatic logic and reversible computing. The latter idea postulates that using adiabatic charging and discharging, all charge taken from the supply during a computation can be put back into the supply afterward by reversing the computation [Svensson’05].

None of these ideas ever made it into real use. Mark Horowitz and his students [Indermanner’94] rightly argued that if one is willing to give up performance, one might as well lower the supply voltage of a traditional CMOS circuit rather than using the more complex adiabatic charging. And the overhead of reversing a computation proved to be substantial and not really practical.

Yet, with voltage scaling becoming harder, the idea of adiabatic charging is gaining some following again. An excellent example of this is shown in this slide, where adiabatic charging is
Adiabatic Logic and Energy Recovery

- Concept explored in the 1990s
  - Proven to be ineffective at that time
- With voltage scaling getting harder, may become attractive again

Example: Resonant Adiabatic Mixed-Signal Processor Array for Charge-Based Pattern Recognition

Adiabatic logic modeled as transmission gate driving capacitive load from resonant clock

Adiabatic mixed-signal multiple-accumulation (MAC). Charge-coupled MOS pair represents variable capacitive load.

[Ref: R. Karakevit, JSSC/87]

At the time of writing, it is hard to see whether the adiabatic approach is limited to niche circuits, or if it may lead to something more substantial.

Self-timed and Asynchronous Logic

- Synchronicity performs best under deterministic conditions and when duty cycle is high
- However, worst-case model does not fair well when variability is high
- In ideal case, self-timed logic operates at "average conditions"

Delay distribution as a function of variability

- Protocol and signaling overhead of self-timed logic made it unattractive when delay distributions were narrow
- This is no longer the case, especially under ultra low-voltage conditions
  - Effective "synchronous island" size is shrinking
- The "design flow" argument does not really hold either
  - Example: Handshake Solutions [Ref: Handshake]

One topic that always floats to the foreground when discussing low-power technology of the future is the potential of asynchronous-timing or self-timing strategies. The belief that self-timing may help to reduce power dissipation is correct, but the reasoning behind it is often misguided. The common understanding is that the power savings come from the elimination of the global clock. In reality, that is not the case. A well-thought-out synchronous-clocking strategy using hierarchical clock gating can be just as efficient, and eliminates the overhead of completion signal generation and protocol signaling. More meaningful is that self-timed methodologies inherently support the “better-than-worst-case” design strategy we are advocating. Variations in activity and implementation platforms are automatically accounted for. In fact, asynchronous design, when implemented in full, realizes an “average case” scenario. In a design world where variations are prominent, the small overheads of the self-timing are easily offset by the efficiency gains.
Yet, this does not mean that all designs of tomorrow will be asynchronous. For smaller modules, the efficiency of a clocked strategy is still hard to beat. This explains the current popularity of the GALS (globally asynchronous locally synchronous) approach, in which islands of synchronicity communicate over asynchronous networks [Chapiro84]. Over time, the size of the islands is bound to shrink gradually.

Another often-voiced concern is that asynchronous design is complex and not compatible with contemporary design flows. Again, this is only a partial truth. The design technology is well-understood, and has been published at length in the literature. The real challenge is to convince the EDA companies and the large design houses that asynchronous design is a viable alternative, and that the effort needed to incorporate the concept into the traditional environments is relatively small. This rather falls into the domains of policy and economic decision-making rather than technology development.

---

**Exploring the Unknown—Alternative Computational Models**

Humans

10–15% of terrestrial animal biomass
10⁷ Neurons/"node"
Since 10⁷ years ago

Ants

10–15% of terrestrial animal biomass
10⁷ Neurons/"node"
Since 10⁷ years ago

Easier to "make" ants than humans
"Small, simple, swarm"

[Courtesy: D. Petrovic, UCB]

---

**Slide 13.11**

We would like to devote the last pages of this book to some far-out speculation. In the text, we have advocated a number of strategies such as concurrency, better-than-worst-case design and aggressive deployment. Although these techniques can be accommodated within the traditional computational models of today, it is clear that doing so is not trivial. To state it simply, these models were not built to deal effectively with statistics.

There are other computational systems that do this much better, famously, those that we encounter in biology and nature. Look, for instance, at the brain, which performs amazingly well under very low signal-to-noise conditions and adapts effectively to failure and changing conditions. Maybe some of the techniques that nature uses to perform computation and/or communication could help us to make the integrated systems of tomorrow work better and more efficiently.

Let us take the case of concurrency, for example. We have discussed earlier how multi- and many-core systems are being adopted as a means to improve performance of SoCs while keeping energy efficiency constant. We can take this one step further. What is it that keeps us from envisioning chips with millions of processors, each of them very simple? This model is indeed working very well in nature – think again about the brain, or alternatively a colony of ants. Instead of building a system based on a small number of very reliable and complex components, a complex reliable system can emerge from the communication between huge numbers of simple nodes. The motto is “small, simple, swarm”.
Slide 13.12
The advantage of these “collaborative” networks is that they avoid the Achilles’ heels of traditional computing, as redundancy is inherent, and, as such, the networks are naturally robust. This may allow for computation and communication components to operate much closer to the von Neumann and Shannon bounds.

Slide 13.13
These ideas are definitely not new. Cellular automata are an example of systems that are built on similar ideas. The “neural networks” concept of the late 1980s is another one—however, that concept was doomed by the limitations of the computational model, the attempt to transplant computational models between incompatible platforms, and the hype. Hence, it has proven to be useful in only a very limited way.

Yet, the idea of building complex electronic systems by combining many simple and non-ideal components found its true calling in the concept of wireless sensor networks, which emerged in the late 1990s. There it was realized that the statistical nature of the network actually helped to create robust and reliable systems, even if individual nodes failed or ran out of power and in the presence of communication failures. In one landmark paper, it was shown how the combination of pure stochastic communications and network coding leads to absolutely reliable systems if the number of nodes is large enough.
estimation algorithm. The technique obviously only works effectively if the estimations are non-correlated, yet have the same mean value.

This technique is inherently robust – failure in one or more of the sensors does not doom the final results, it just impacts the signal-to-noise ratio or the QOS (quality-of-service). As such, aggressive low-energy computation and communication techniques can be used. The only block that needs to be failproof is the fusion module (and even that can be avoided).

An attentive reader may notice that the SNOC approach is nothing less than an extension of the “aggressive deployment” approach advocated in Chapter 10. Consider, for instance, the example of Slide 10.53, where a computational block was supplemented by an estimator to catch and correct the occasional errors. The SNOC simply eliminates the computational block altogether, and uses only estimators, realizing that the computational block is nothing more than a complex estimator itself.

Slide 13.14
Based on these observations, a number of researchers have started to explore the idea of bringing similar ideas to the chip level – the “sensor-network-on-a-chip (SNOC)” approach [Narayanan’07]. Instead of having a single unit perform a computation (such as filtering or coding), why not let N simple units simultaneously estimate the result and have a single fusion block combine these estimations into a final output. Each of the “sensors” only works on a subset of data and uses a simplified
Example: PN-Code Acquisition for CDMA

- Statistically similar decomposition of function for distributed sensor-based computation.
- Robust statistics framework for design of fusion block.
- Creates better result with power savings of up to 40% for 8 sensors in PN-code acquisition in CDMA systems.
- New applications in filtering, ME, DCT, FFT, and others.

[Ref: S. Narayanan, Asilomar’07]

and a sub-set of the pn-code. The fusion block either accumulates over the different sensors or performs a median filtering. As can be observed, the algorithm performs very well even in the presence of large number of failures. In addition, owing to aggressive deployment, the energy efficiency is improved by 40%.

One can envision many examples that fall into the same category. In fact, any application in the RMS (recognition, mining, synthesis) class is potentially amenable to the ideas presented. With the growing importance of these applications, it is clear that the opportunities for innovation are huge.

Book Summary

- Energy Efficiency one of the (if not the most) compelling issues in integrated-circuit design today and in the coming decades
- The field has matured substantially
  - From “getting rid of the fat” and reducing waste
  - To “truly energy-lean” design technologies
- Still plenty of opportunities to move beyond what can be done today
  - There is plenty of room at the bottom

Slide 13.15
An example of an SNOC is shown in this slide. The application is pn-code acquisition, an essential function in any wideband-CDMA wireless receiver. Its main task is to correlate an incoming data stream with a long pseudo-random code. The traditional approach is to use a single correlator – which obviously is vulnerable to failures. As shown in the chart, a single failure dooms the hardware. The SNOC-architecture divides the function over many (16–256) simple correlators, each operating on a sub-sampled data stream.

Slide 13.16
The goal of this book was to present low-power design in a methodological and structured fashion. It is our hope that, by doing so, we have offered you the tools to engage effectively in state-of-the-art low-energy design, and, furthermore, to contribute actively to the field in the future.
Interesting References for Further Contemplation

Books and Book Chapters

Articles
Index

A
Abstraction design methodology, 79–80
Abstraction levels, optimizations at, 114
Accelerator approach, 143
Active (dynamic) power, 54–55
Active deskew, 212
Adaptive body bias (ABB), in runtime optimization, 265–268
  advantage at low $V_{DD}/V_{TH}$, 269
Adiabatic charging approach, 163–164
Adiabatic logic, 350
Aggressive deployment (AD), in runtime optimization, 272
  algorithmic-level AD, effectiveness, 279
  components, 273
    error correction, 273
    error detection, 273
  voltage-setting mechanism, 273
Air conditioning system, power issues in, 3
Algebraic transformations, 101
Algorithmic BTWC, 278
Alpha power law model, nanometer transistors, 29
Alpha-power based delay model, 84
Amdahl's law, 145–146
Application-specific instruction processors (ASIPs), 136, 141
  advantage, 141
Application-specific integrated circuit (ASIC) design, 97–98
Architecture, algorithms and systems level optimizations
  @ design time, 113–148
  in 1990s, 121–122
  architectural choices, 137
  concurrency exploitation, 116–118
    See also Concurrence
  (re)configurable processors, 143
  design abstraction stack, 115
  domain of video processing, 141–142
  embedded applications, 146
  energy-delay space mapping, 119
  extensible-processor approach, 141–142
  flexibility, 135
    quantifying flexibility, 135
    trade-off between energy efficiency and, 136–137
  hardware accelerators, 142–144
  locality of reference, 132
  matching computation to architecture issue, 128–129
  to minimize activity, 133–134
  multi-antenna techniques, 129–130
parallel implementation, 117
pipelining, 117–118
programming in space versus time, 144
simple versus complex processors, 138
software optimizations, 133
time-multiplexing, 120
word-length optimization, 129–131
See also Platform-based design strategy; Singular-value
decomposition (SVD)
Asynchronous logic, 350–351
Asynchronous signaling, 166
Automated optimization, 81

B
Back-gated (BG) MOSFET, 203–204
Battery technology
  battery storage, as limiting factor, 6
  energy storage, calculation, 9
  evolution, 7
  fuel cells, 8–9
  higher energy density, need for, 8
  Lithium-ion, 7
  micro batteries, 9
  saturating, 7
  supercapacitor, 10
Better-than-worst-case (BTWC) design, 272
Biological machinery, 13
Bipolar systems, heat flux in, 14–15
6T Bitcells, 202
Bitline leakage, SRAM
during read access, 196
solutions, 197
Body biasing
  body bias circuitry, 226
    central bias generator (CBG), 226
    local bias generators (LBGs), 226
  nanometer transistors, 35
    body-effect parameter $\gamma$, 30
    forward, 30–31
    reverse, 30–31
in standby mode leakage control, 224–225
in standby mode leakage reduction of embedded
  SRAM, 244
  body biasing and voltage scaling, combining, 245
Body biasing (cont.)
  forward body bias (FBB), 245
  raised VSS and RBB, combining, 245
  reverse body bias (RBB), 244

See also Adaptive body bias (ABB)

Boolean–von Neumann–Turing principles, 347
Boosted-gate MOS (BGMOS), in standby mode leakage control, 218–219
Boosted-sleep MOS, in standby mode leakage control, 219
Bus protocols and energy, 172
Bus-invert coding (BIC), 168–169

C
Canary-based feedback mechanism, 243

Capacitors
  See also Charging capacitors
Carbon-nanotube (CNT) transistors, 51
Cell writability, SRAM, 191
Central bias generator (CBG), 226
Channel length impact on nanometer transistors threshold voltages, 31
Channel-coding techniques, 167
Charge recycling, 164–165
Charging capacitors, 55, 57
  driving from a constant current source, 57
  driving using sine wave, 58
Chip architecture and power density, 4
Circuit optimization, 83–84, 114–115
Circuit with dc bias currents, 70
  power management, 70
  trade off performance for current, 70
Circuit-level techniques, optimizing power @ design time, 77–111
  abstraction methodology, 79–80
  algebraic transformations, 101
  alpha-power based delay model, 84
  ALU for 64-bit microprocessor, 96
  circuit optimization framework, 83
    generic network, 84
    complex gates, 108
    delay modeling, 84
    ‘design-time’ design techniques, 78
  dual-VTH domino, 106
    for high-performance design, 107
  dynamic energy, 85
  dynamic-power optimization, 78
  energy–delay trade-off optimization, framework, 78–79
  hierarchy methodology, 79–80
  inverter chain, 86–88
  layered approach, 80
  leakage at design time, 102
    reducing the voltage, 103
    reducing using higher thresholds, 103
    reducing using longer transistors, 103
    reducing, 103–104
  for leakage reduction, 199
  level-converting flip-flops (LCFFs), 95
  logical effort formulation, 85
  longer channels, 104
  low-swing bus and level converter, 96
  multiple thresholds voltages, 104–106
  optimal energy–delay curve, 79, 82
  optimization model, refining, 103
    leakage energy, 103
    switching energy, 103
  reducing active energy @ design time, 82
  Return on Investment (ROI), optimizing, 86
  ‘run-time’ optimizations, 78
  shared n-well, 94
  sizing, transistor, 97–98
    continuous, 98
    discrete, 98
  static power optimization, 78
  technology mapping, 98–100
    logical optimizations, 100
    variables to adjust, 79
    continuous variables, 79, 80–81
    discrete variables, 79
  See also Multiple supply voltages

Circuits and systems, optimizing power @ runtime, 249–288
  adaptive body bias, 267–268
  aggressive deployment at the algorithm level, 278
  clock frequency, adjusting, 252
  disadvantages, 252
  converter loop sets VDD, fclk, 260
  delay sensitivity, 265
  dynamic frequency scaling (DFS), 253
  dynamic logic, 263
  dynamic voltage scaling (DVS), 253–254
  energy–performance characteristics, 277
  error rate versus supply voltage, 274
  generalized self-adapting approach, 271
  high-performance processor at low energy, 261
  relative timing variation, 264
  static CMOS logic, 263
  stream-based processing and voltage dithering, 256
  threshold variability and performance, 266
  timing, managing, 281
  using discrete voltage levels, 255
  variable workload
    adapting to, 252
      in general-purpose computing, 251
      in media processing, 251
  VDD and fclk, relating, 257
    on-line speed estimation, 257
    self-timed, 257
    synchronous approach, 257
    table look-up, 257
  VDD and throughput, 253
  voltage scheduling impact, 260
  voltage/frequency scheduling, impact, 259
  See also Aggressive deployment (AD)

Circuits and systems, optimizing power @ standby, 207–230

See also Standby mode

Circuits with reduced swing, 56
  Circuit-switched versus packet-based network, 175
  Clock distribution network, optimizing power @ design time, 178–180
    advantages, 178
Index

reduced-swing clock distribution, 178–179
transmission line based, 179
Clock frequency, adjustment, 252
Clock gating, 209–210
implementing, 210
low power design flow, 327
clock-gating insertion, 328
conventional RTL code, 328
data gating, 329
 glitchfree verilog, 329
global clock gating, 328
instantiated clock-gating cell, 328
local clock gating, 328
low power clock-gated RTL code, 328
verilog code, 328
reducing power, 210
Clock hierarchy, 211
Closed-loop feedback approach, for DRV, 242
Clustered voltage scaling (CVS), 94
CMOS systems
heat flux in, 14–15, 17
power dissipation in, 54
active (dynamic) power, 54
static (leakage) power, 54
static currents, 54
reducing SRAM power, 203–204
Code transformations, 125–126
Cede-division multiple access (CDMA), 136–137
Cede-division multiplexing (CDM), 171
Coding strategies, 167
activity reduction through, 168
bus-invert coding (BIC), 168–169
channel-coding techniques, 167
error-correcting coding (ECC), 167, 170
source codes, 168
transition coding techniques, 169–170
Collaborative networks, 352
Communication infrastructure, power consumption in, 2–3
Complex gates, 108
complex-versus-simple gate trade-off, 109
Complex logic, power dissipation evaluation in, 62
Composite current source (CCS), 320
Computation, power consumption in, 2–3
Computational architecture, 128–129
Computational efficiency, improving, 128
Computing density, 19
Concerns over power consumption, 1–2
Concurrency, optimization using, 116–118, 346
in 2000s, 122–123
alternative topologies, 126
concurrent compilers to pick up the pace, 125
fixed EOP, 119
fixed performance, 119
manipulating through transformations, 124
and multiplexing combined, 120
quest for, 123
Conditional probability, 61
(re)configurable processors, 143
Constant-current (CC) technique, 30
Constraints, design, see Design constraints
Consumer and computing devices, 5–6
‘Microwatt nodes’, 6
‘Milliwatt nodes’, 5
‘Watt nodes’, 5–6
Continuous design parameters, 79–81
Cooling issues
chip cooling system, 3–4
computing server rack, 3
‘Custom voltage assignment’ approach, 93

D
Data gating, low power design flow, 329
data-gating insertion, 330
logic synthesizer, 330
RTL code, 330
data-gating verilog code, operand isolation, 330
conventional code, 330
low power code, 330
low power version, 329
Data link/media access layer, 167
Data retention voltage (DRV), in embedded SRAM,
236–237
approaching, 242
closed-loop feedback approach, 242
open-loop approach, 242
lowering using error-correcting (ECC) strategies, 240
power savings of, 237
process balance impact, 238
process variations impact on, 238–239
reducing, 240
by lowering voltage below worst-case value, 240
optimization, 240
statistical distribution, 239–240
and transistor sizes, 237
Decoupling caps
effectiveness, 342
placement, 220
voltage drop mitigation with, 342
Delay (s), 54
Delay modeling, 84
Delay sensitivity, 265
Design abstraction stack, 115
Design constraints, 2–3, 5
communication infrastructure, growth, 2–3
computation, growth, 2–3
cooling issues, 3
mobile electronics emergence, 5
‘zero-power electronics’ emergence, 10–11
Design phase analysis methodology, low power design, 323
Design phase low power design, 327
clock gating, 327
data gating, 327
f\text{\text{eff}} minimizing, 327
memory system design, 327
‘Design time’ design techniques, 78
Design time techniques, in standby mode leakage reduction
of embedded SRAM, 235
Differential logic networks, 61
Differential signaling, 162
Digital frequency dividers (DFDs), 211
Digital signal processors (DSPs), 132, 139–140
Digital signal processors (DSPs) (cont.)
advantages, 140
performance of, 140
Diode reverse-bias currents, 69
Direct-oxide tunneling currents, nanometer MOS
transistor, 39
Disappearing electronics, see Zero-power electronics
Discrete design parameters, 79–81
Discrete voltage levels, 255
Dissipation, power, l
Dithering, 255
Double-gated (DG) MOSFET, 203–204
Double-gated fully depleted SOI, 49
Drain induced barrier lowering (DIBL), 29, 32, 35
Drain leakage, 66
Drain-induced barrier lowering (DIBL) effect, 235
Dual voltage approach, 93
Dynamic body biasing (DBB), 266
dynamics of, 225–226
effectiveness of, 227
in standby mode leakage control, 224–227
Dynamic energy, 85
Dynamic frequency scaling (DFS), 253
Dynamic hazards, 63
Dynamic logic networks, 60–61, 263
Dynamic power, 19, 53
consumption, 58
dissipation, 70
reduction by cell resizing, 332–333
in standby, 208
Dynamic RAM (DRAM), 183–205
Dynamic voltage and frequency scaling
(DVFS), 325
Dynamic voltage drop, 339, 341, 343
Dynamic voltage scaling (DVS), in runtime optimization,
253–254
ABB and, combining, 269–271
in general-purpose processing, 259
verification challenge, 262
workload estimation, 255
Dynamic-power optimization, 78

E
Edge-triggered flip-flop, 95
Effective capacitance, 59
Effective resistance check, 339
Embedded SRAM, 234
standby leakage reduction, 234
body biasing, 244
canary-based feedback mechanism, 243
data retention voltage (DRV), 236–237
design-time techniques, 235
leakage current reduction, 235
periphery leakage breakdown, 246
by raising $V_{SS}$, 243–244
voltage knobs, 235
voltage reduction, 235
voltage scaling approach, 235–236
voltage scaling in and around the bit-cell, 246
Energy (Joule), 54
Energy efficiency of brain, 13
Energy per operation (EOP), 116–117, 290
minimum EOP, 294–295
Energy recovery, 350
Energy scavenging, 12
Energy storage technologies, 9–10
See also Battery technology
Energy-area-delay tradeoff in SVD, 131
Energy–delay ($E–D$) trade-off optimization
framework, 78–79
optimal energy–delay curve, 79
Energy–Delay space, 54, 73–74, 119
Equivalent oxide thickness (EOT), 40
Error correction, aggressive deployment (AD), 273
Error detection, aggressive deployment (AD), 273
Error rate versus supply voltage, runtime optimization, 274
Error-correcting coding (ECC) strategies, 167, 170
combining cell optimization and, 241–242
DRV lowering using, 240

F
Factoring, 101
Fast Fourier Transform (FFT) module, 297–298
energy-performance curves, 298
sub-threshold FFT, 299
Fine-grained power gating, 335
FinFETs, 49
backgated FinFET, 50
Fixed deskew, 212
Fixed-voltage scaling model, 16
Flit-routing, 176
Forced transistor stacking, in standby mode leakage control,
215–216
Fowler–Nordheim (FN) tunneling, 38
Frequency scaling model, 72
Frequency-division multiplexing (FDM), 171
Fuel cells, 8–9
miniature fuel cells, 9
Full-depleted SOI (FD-SOI), 19, 48

G
Gate leakage, 66
nanometer MOS transistor, 37–38
gate leakage current density limit versus simulated
gate leakage, 41
gate-induced drain leakage (GIDL), 36–37
mechanisms, 38
Gate tunneling, 69
Gate-induced drain leakage (GIDL), 235
Gate-level trade-offs for power, 99–100
Generalized low power design flow, 321
design phase, 321
implementation, 321
RTL design, 321
system-level design (SLD), 321
Glitch-free verilog code, clock gating
low power design flow, 329
prevention latch, 329
Glitching
Index

occurrence, 64–65
in static CMOS, 63
Global clock gating, 328
Globally asynchronous locally synchronous (GALS) methodology, 166, 351

H
Hardware accelerators, 142–144
Heterogeneous networking topology, 174
Hetero-junction devices, 47
Hierarchical bitlines, SRAM, 195
Hierarchical optimization, challenge in, 114
Hierarchical wordline architecture, SRAM, 195
Hierarchy design methodology, 79–80
High-performance microprocessors, 3–4
Homogeneous networking topology, 174
binary tree network, 174
crossbar, 174
mesh, 174
Human brain, power consumption by, 13

I
Idealized wire-scaling model, 153
Implementation phase low power design, 323, 331–332
low power synthesis, 332
multiple supply voltages, 332
power gating, 332
power integrity design, 332
slack redistribution, 332
Instruction loop buffer (ILB), 132
Integrated chips (ICs), memory role in, 184
integrated clock gating cell, 329
Integrated power converter for sensor networks, 284
Interconnect network/Interconnect-optimization, @ design time, 151–180
charge recycling, 164
circuit-switched versus packet-based, 175
communication dominant part of power budget, 153
data link/media access layer, 167
idealized wire-scaling model, 153
increasing impact of, 152
interconnect scaling, 156
ITRS projections, 152
layered approach, 157
physical layer, 158
repeater insertion, 158
logic scaling, 155
multi-dimensional optimization, 160
networking topologies, 174
binary tree network, 174
crossbar, 174
exploration, 175
heterogeneous, 174
hierarchy, 174
homogeneous, 174
mesh, 174
network trade-offs, 173
OSI approach, 151
OSI protocol stack, 157–158
quasi-adiabatic charging, 164
reduced swing, 160
reducing interconnect power/energy, 157
research in, 154–155
dielectrics with lower permittivity, 154
interconnect materials with lower resistance, 154
novel interconnect media, 15
shorter wire lengths, 155
signaling protocols, 166
wire energy delay trade-off, 159–160
See also Clock distribution network
Interconnect scaling, 156
Inverted-clustering, 175
Inverter chain, 86–87
gate sizing, 87
VDD optimization, 87

J
Junction leakage, 66

K
Kogge–Stone tree adder, 88–89, 109, 274, 276
sizing vs. dual-VDD optimization, 89

L
Latch-retaining state during sleep, 222–223
Layered design methodology, 80
LC-based DC–DC (buck) converter, 284
Leakage, 66, 346
advantages, 102
components, nanometer MOS transistor, 33
drain leakage, 66
at design time, 102
effects/concerns, 18, 20, 21, 37
gate leakage, 66
in standby mode, 214
See also under Standby mode
junction leakage, 66
mechanisms, memory and, 192
See also under Memory
power reduction by VTH assignment, 332–333
reduced threshold voltages, 34
sub-threshold leakage, 33, 35, 67
Level-converting flip-flops (LCFFs), 95
pulsed LCFF, dynamic realization, 95
pulsed precharge (PPR) LCFF, 95
Liberty power, 319
Lithium-ion battery technology, 7–8
Local bias generators (LBGs), 226
Local clock gating, 328
Locality of reference, 132
Logic function, 59
Logic networks
activity as a function of topology, 60
differential logic networks, 61
See also Complex logic; Dynamic logic
Logic scaling, 155
Logical effort based design optimization methodology, 71
Logical optimizations, 100
log ic restructuring, 100
Logical effort formulation, 85
Logic-sizing considerations, sub-threshold design, 300
Loop transformations, 124
Low power design methodologies and flows, 317–344
clock gating, 327
design phase, 327
analysis methodology, 323
dynamic voltage drop impact, 343
in implementation phase, 323, 331–332
methodology issues, 318
power analysis, 318
power integrity, 318
power reduction, 318
motivations, 318
minimize effort, 318
minimize power, 318
minimize time, 318
power analysis methodology, 321
issues, 322
method, 321
motivation, 321
over project duration, 324
power characterization and modeling issue, 318–320
SPICE-like simulations, 320
state-dependent leakage models, 320
state-independent leakage models, 320
power-down modes, 325
power integrity methodologies, 339
slack redistribution, 332
system phase analysis methodology, 322, 324–326
challenges, 324
f_{eff} minimization, 324
modes, 324
parallelism, 324
pipelining, 324
V_{DD} minimization, 324
voltage drop mitigation with decoupling caps, 342
See also Clock gating; Generalized low power design flow;
Memory system design; Multi-V_{DD}; Power gating

\textbf{M}
\begin{itemize}
\item Magnetoresistive RAM (MRAM), 183–205
\item MATLAB program, 81
\item Media access control (MAC), 167, 170–171
\end{itemize}
Memory, optimizing power @ design time, 183–205
cell array power, 192
leakage and, 192
multiple threshold voltages reducing, 193
multiple voltages, 194
sub-threshold leakage, 192
threshold voltage to reduce, 192
low-swing write, 201
processor area dominated by, 184
role in ICs, 184
structures, 185
power for read access, 185
power for write access, 185
power in the cell array, 185
See also Static random access memory (SRAM)
Memory, optimizing power @ standby, 233–247
processor area dominated by, 234
See also Embedded SRAM
Memory system design, low power, 330–331
objectives
C_{eff} minimization, 331
challenges, 331
f_{eff} minimization, 331
power reduction methods, 331
trade-offs, 331
split memory access, 331
Metrics, 54
delay (s), 54
energy (Joule), 54
energy delay, 54
power (Watt), 54
propagation delay, 54
Micro batteries, 9
Micro-electromechanical systems (MEMS), 51, 348
'Microwatt nodes', 6
'Milliwatt nodes', 5
Minimum energy point, moving, 313
from sub-threshold to moderate inversion, 312–313
using different logic styles, 312
using switching devices, 312
Mobile electronics emergence, as design constraint, 5
Mobile functionality limited by energy budget, 6
Moore's law, 6, 18
Motes, 10
MTCMOS derivatives, in standby mode state loss prevention, 223
Multi-dimensional optimization, interconnect, 160
Multiple supply voltages, 90–92
block-level supply assignment, 90–91
conventional, 93
distributing, 93
multiple supplies inside a block, 90–91
Multiple threshold voltages
in power optimization, 104
reducing SRAM leakage, 193
Multiple-input and multiple-output (MIMO) communications, SVD processor
tor, 129–130
Multi-V_{DD}, low power design, 337–338
flow, 338
issues, 338
level shifters, 338
partitioning, 338
physical design, 338
timing verification, 338
voltages, 338
\textbf{N}
\begin{itemize}
\item NAND gates, 60, 64
\item Nano-electromechanical systems (NEMS), 348
\item Nanometer transistors, 25–52
\item advantages, 26
\end{itemize}
INDEX

alpha power law model, 29
behavior, 25–26
body bias, forward and reverse, 30
challenging low power design, 26
device and technology innovations, 45–46
strained silicon, 46–47
DIBL effect, 32, 35
drain current under velocity saturation, 27
FinFETs, 49–50
leakage components, 33
direct-oxide tunneling currents, 39
high-k dielectrics, 40
high-k gate dielectric, 39
leakage effects, 37
reduced threshold voltages impact on, 34
sub-threshold current, 35
sub-threshold leakage, 33, 35
temperature sensitivity and, 41
See also Gate leakage
output resistance, 29
65 nm bulk NMOS transistor, $I_D$ versus $V_{DS}$ for, 27
Silicon-on-Insulator (SOI), 48
See also individual entry
sub-100 nm transistor, 26–27
their models and, 25–52
sub-100 nm CMOS transistors
threshold control, evolution, 31
threshold voltages, channel length impact on, 31–32
thresholds and sub-threshold current, 30
variability impact, 42–43
environmental source, 43
physical source, 43
process variations, 44
threshold variations, 45
variability sources and their time scales, 43
variability sources, 43
Nanotechnology, 348
nano-mechanical relays, 348
Need for power, 2
Negative bias temperature instability (NBTI), 44
Network trade-offs, 173
Network-on-a-chip (NoC), 166, 172–173, 176
Neumann von and Shannon bounds, 352
Neural networks concept, 352
Non-traditional bit-cells, 202
Novel switching devices, 347
20 nm technology, 21
65 nm bulk NMOS transistor, $I_D$ versus $V_{DS}$ for, 27

P
Packet-switched networks, 175–176
Parallelism, 117, 324, 326
Pareto-optimal curve, 82
Partially-depleted (PD-SOI), 48
Pass-transistor logic (PTL) approach, 313
leakage control in, 314
Periphery leakage breakdown, embedded SRAM, 246
Physical layer of interconnect stack, 158
Pipelining, 117–118, 324, 326
Platform-based design strategy, 146
heterogeneous platform, 147
NXP Nexperia™ platform, 146
OMAP platform™ for Wireless, 147
Pleiades network, 177
PMOS transistors, 65
PN-code acquisition for CDMA, 354
Power (Watt), 54
and delay, relationship between, 73
dissipation, 61–62
in CMOS, 54
distribution, 285–286
power density, 18
versus energy, 14
Power domains (PDs), 280, 346
challenges, 280
interfacing between, 282
in sensor network processor, 282
Power gating
low power design, 335
fine-grained power gating, 335
flow, 337
issues, 336
physical design, 335
switch placement, 335
switch sizing, 336
in standby mode leakage control, 217–218
Power integrity methodologies, 339
dynamic voltage drop, 341
resistance check, 340
static voltage drop, 341
stimulus selection, 340
verification flow, 339
Power manager (PM), 280
Power-down modes, 325
clock frequency control, 325
issues, 325
trade-offs, 325
$V_{DD}$ control, 325
Power-limited technology scaling, 22
Processors, power trends for, 15–17
Propagation delay, 54
of sub-threshold inverter, 295
Pull-up and pull-down networks, for minimum-voltage
operation, 293
Pulsed LCFF, dynamic realization, 95
Pulsed precharge (PPR) LCFF, 95

Q
Quasi-adiabatic charging, 164
R
Random doping fluctuation (RDF), 186
RAZOR, in runtime optimization, 275
distributed pipeline recovery, 276
voltage setting mechanism, 276
Read-assist techniques, for leakage reduction, 199
Read-power reduction techniques, 201–202
Reconvergent fan-out, 61
Reduced-swing circuits, 160–162
issues, 163
Reduced-swing clock distribution, 178–179
Register-transfer level (RTL) code, 209
Relay circuit design and comparison, 349
Return on Investment (ROI), optimizing, 86
Rules, low power design, 345–347
Runtime optimization, 346
‘Run-time’ optimizations, 78

dynamic power reduction by cell resizing, 332–333
leakage power reduction by $V_{TH}$ assignment, 332–333
objective, 332
slack redistribution flows, 334
issues, 334
trade-offs, 334
Sleep mode management, see Standby mode
Software optimizations, 133
Source codes, 168
Spatial programming, 129
Split memory access, low power design methodology, 331
6T SRAM cell, 204
Stack effect, 68
Standby mode, optimizing power @ standby, 207–230
concern over, 209
decoupling capacitor placement, 220
design exploration space, 213
dynamic power in standby, 208
energy consumption in, 209
See also Clock gating
impacting performance, 220–221
in microprocessors and microcontrollers, 213
latch-retaining state during, 222–223
leakage challenge in, 214
control techniques, 215
leakage control
boosted-gate MOS (BGMOS), 218–219
boosted-sleep MOS, 219
dynamic body biasing (DBB), 224–225
forced transistor stacking, 215–216
power-gating technique, 217–218
supply voltage ramping (SVR), 227–228
transistor stacking, 215
virtual supplies, 219
See also under Embedded SRAM
MTCMOS derivatives preventing state loss, 223
preserving state, 222
reaching, 219–220
sizing, 221
sleep modes and sleep time, trade-off between, 212
sleep transistor layout, 224
sleep transistor placement, 223
standard cell layout methodology, integration in, 229
versus active delay, 216
State retention flip-flops (SRFFs), 337
Static (leakage) power, 54–55
Static CMOS logic, 263
glitching in, 63
Static currents, 54–55
Static noise margin (SNM), SRAM, 188–189
cells with pseudo-static SNM removal, 203
lower precharge voltage improving, 198
with scaling, 190
Static power, 53
dissipation, 69–70
drawbacks, 18
Static random access memory (SRAM), 183–205
bit-cell array, 187
power consumption within, 191
BL leakage during read access, 196
cell writability, 191

S
Scaling/Scaling model, 17
direct consequences of, 152–153
fixed-voltage scaling model, 16
frequency, 72
idealized wire-scaling model, 153
interconnect scaling, 156
logic scaling, 155
supply voltage scaling, 17
traditional, 72
Scavenging, energy, 12
Self-adapting approach, 271
Self-adjusting threshold voltage scheme (SATS), 266–267
Self-timing strategies, 350–351
Sense amplifier based pass-transistor logic (SAPTL), 314–315
Sensor network concept, 352
Sensor network-on-a-chip (SNOC), 353
Sensor networks, integrated power converter for, 284
Shannon theorem, 156
Shannon–von Neumann–Landauer limit, 295, 311
Shared n-well, 94
Shared-well technique, 96
Short circuit currents, 63, 65–66
modeling, 66
as capacitor, 66
Silicon-on-Insulator (SOI), 48
double-gated fully depleted SOI, 49
fully-depleted (FD-SOI), 48
partially-depleted (PD-SOI), 48
types of, 48
Simple versus complex processors, 138
Singular-value decomposition (SVD) processor
energy-area-delay tradeoff in, 131
for MIMO, 129–130
optimization techniques, 130–131
power/area optimal 4x4 SVD chip, 131
Sizing, transistor, 97–98
continuous, 98
discrete, 98
Slack redistribution, low power design, 332–333
dynamic & leakage power optimization, 332
See also under Embedded SRAM
MTCMOS derivatives preventing state loss, 223
preserving state, 222
reaching, 219–220
sizing, 221
sleep modes and sleep time, trade-off between, 212
sleep transistor layout, 224
sleep transistor placement, 223
standard cell layout methodology, integration in, 229
versus active delay, 216
State retention flip-flops (SRFFs), 337
Static (leakage) power, 54–55
Static CMOS logic, 263
glitching in, 63
Static currents, 54–55
Static noise margin (SNM), SRAM, 188–189
cells with pseudo-static SNM removal, 203
lower precharge voltage improving, 198
with scaling, 190
Static power, 53
dissipation, 69–70
drawbacks, 18
Static random access memory (SRAM), 183–205
bit-cell array, 187
power consumption within, 191
BL leakage during read access, 196
cell writability, 191
Index

data retention voltage (DRV), 188
embedded SRAM, 234
functionality constraint in, 205
hierarchical bitlines, 195
hierarchical wordline architecture, 195
metrics, 186
  area, 186
  functionality, 186
  hold, functionality metric, 188
  power, 186
  read, functionality metric, 188
  write, functionality metric, 188
power breakdown during read, 194–195
power consumption in, 186–187
process noise margin, 189
6T SRAM cell, 204
static noise margin (SNM), 188–189
sub-threshold SRAM, 303
topology, 187
V\textsubscript{DD} scaling, 198
voltage transfer characteristic (VTC) curves, 189
write margin, 190
write, power breakdown during, 199–200
alternative bit-cell reducing, 200
charge recycling to reduce write power, 200–201
Static routing, 175
Static voltage drop, 339, 341
Static power optimization, 78
Statistical computational models, 347
Strained silicon concept, 46–47
Stream-based processing, 256
Sub-100 nm CMOS transistors, 26–27
models for, 28
simplification in, 28
Substrate current body effect (SCBE), 29
Sub-threshold current, nanometer MOS transistor, 30, 35
as a function of V\textsubscript{TBD}, 36
Sub-threshold design challenges in, 299
data dependencies impact, 301
erratic behavior, 302
logic-sizing considerations, 300
modeling in moderate-inversion region, 306–307
process variations impact, 300
read current/bitline leakage, 302
SNM, 302
soft errors, 302
timing variance, 305
write margin, 302
CMOS inverter, 292
microprocessor, 304
moving away the minimum energy point from, 312
power dissipation of, 296
propagation delay of, 295
prototype implementation of, 304
SRAM cell, 303
sub-threshold FFT, 299
sub-V\textsubscript{TBD} memory, 302
Sub-threshold leakage, 17, 67, 192
nanometer MOS transistor, 33
Supercapacitor, 10
Supply and threshold voltage trends, 20
Supply voltage ramping (SVR), in standby mode leakage control, 227–228
impact of, 228
Supply voltage scaling, 17, 82–83
Suspended gate MOSFET (SG-MOS), 51
Switch sizing, power gating, 336
Switched-capacitor (SC) converter concept, 283–284
Synchronous approach, 257
System-level design (SLD), in generalized low power design flow, 321
System-on-a-chip (SoC), 165
System phase analysis methodology, 322, 324–325
See also under Low power design
System phase low power design flow, 326
Systems-on-a-Chip (SoC), 3
  complications, 20
T
Technology generations, power evolution over, 15
Technology mapping, in optimizations, 98–100
Temperature gradients and performance, 4
Temperature influence, nanometer transistors leakage, 41
Temporal correlations, 63–64
Thermal voltage, 34
Threshold control, nanometer transistors, 31
Threshold current, nanometer transistors, 30
Threshold variations, nanometer transistors, 45
Threshold voltages
  exploitation, 161
  nanometer transistors, channel length impact on, 31–32
Time-multiplexing, 120
Timing, managing, 281
  basic scheduling schemes, 281
  metrics, 281
Trade-off, 78–79
Transistor stacking, in standby mode leakage control, 215
8T transistor, 203
Transition probabilities for basic gates, 60
Transition coding techniques, 169–170
Transmission line based clock distribution network, 179
Tree adder, 88–89
  in energy–delay space, 89–90
  multi-dimensional search, 90
See also Kogge–Stone tree adder
U
Ultra low power (ULP)/voltage design, 289–316, 347
complex versus simple gates, 312–313
dynamic behavior, 296
EKV model, 309–310
energy–delay trade-off, 315
high-activity scenario, 309
low-activity scenario, 309
minimum energy per operation, 294–295
minimum operational voltage of inverter, 291
  pull-up and pull-down networks for, 293
  modeling energy, 308
  opportunities for, 290–291
Ultra low power (ULP)/voltage design (cont.)
  power–delay product and energy–delay, 297
  PTL, leakage control in, 314
  sense amplifier based pass-transistor logic (SAPTL), 314–315
  size, optimizing over, 310
  $V_{DD}$, optimizing over, 310
  $V_{TH}$, optimizing over, 310
  See also Fast Fourier Transform (FFT) module;
    Sub-threshold design
  Ultracapacitor, 10

V
  Variability impacting nanometer transistors leakage, 42
    See also under Nanometer transistors
  $V_{DD}$ scaling, 198–199
  Velocity saturation effect, 27
    drain current under, 27
  Verilog code, clock gating, 328
    low power design flow
      conventional RTL code, 328
      instantiated clock-gating cell, 328
      low power clock-gated RTL code, 328
  Video, optimizing energy in, 141–142
  Virtual Mobile Engine (VME), 145
  Virtual supplies, in standby mode leakage control, 219
  ‘Virtual’ tapering, 87
  Voltage dithering, 255–256

W
  ‘Watt nodes’, 5–6
  Wave-division multiplexing (WDM), 171
  Weak inversion mode, 292
  Wire energy delay trade-off, interconnect, 159
  Wireless sensor networks (WSN), 10–11
  Word length optimization, 129–131
  Write margin, SRAM, 190
  Write power saving approaches, 201–202

X
  XOR gates, 60

Z
  ‘Zero-power electronics’ emergence, 10–11
Continued from page ii

Routing Congestion in VLSI Circuits: Estimation and Optimization
Prashant Saxena, Rupesh S. Shelar, Sachin Sapatnekar

Ultra-Low Power Wireless Technologies for Sensor Networks
Brian Otis and Jan Rabaey

Sub-Threshold Design for Ultra Low-Power Systems
Alice Wang, Benton H. Calhoun and Anantha Chandrakasan

High Performance Energy Efficient Microprocessor Design
Vojin Oklobdzija and Ram Krishnamurthy (Eds.)

Abstraction Refinement for Large Scale Model Checking
Chao Wang, Gary D. Hachtel, and Fabio Somenzi

A Practical Introduction to PSL
Cindy Eisner and Dana Fisman

Thermal and Power Management of Integrated Systems
Arman Vassighi and Manoj Sachdev

Leakage in Nanometer CMOS Technologies
Siva G. Narendra and Anantha Chandrakasan

Statistical Analysis and Optimization for VLSI: Timing and Power
Ashish Srivastava, Dennis Sylvester, and David Blaauw