In this chapter, we discuss techniques for optimizing power in memory circuits. Specifically, we focus on embedded static random access memory (SRAM). Though other memory structures such as dynamic RAM (DRAM), Flash, and Magnetoresistive RAM (MRAM) also require power optimization, embedded SRAM is definitely the workhorse for on-chip data storage owing to its robust operation, high speed, and low power consumption relative to other options. Also, SRAM is fully compatible with standard CMOS processes, whereas the other memory options are technology-based solutions that usually require special tweaks to the manufacturing process (e.g., special capacitors for embedded DRAM).

This chapter focuses on design time approaches to reducing power consumption for an active SRAM. Although most of the cells in a large SRAM are not accessed at any given time, they must remain in a state of alert, so to speak, to provide timely access when required. This means that the total active power of the SRAM consists of both the switching power of the active cells and the leakage power of the non-active cells of the SRAM.
Role of Memory in ICs

- Memory is very important
- Focus in this chapter is embedded memory
- Percentage of area going to memory is increasing

Slide 7.2
Almost all integrated chips of any substantial complexity require some form of embedded memory. This frequently means SRAM blocks. Some of these blocks can be quite large. The graphs on this slide establish a trend that the majority of processor area in scaled technologies is dedicated to SRAM cache. As higher levels of the cache hierarchy move on-chip, the fraction of die area consumed by SRAM will continue to balloon. Though large caches dominate the area, many recent processors and SoCs contain dozens or even hundreds of small SRAM arrays used for a variety of different purposes. From this, the importance of memory to the functionality and area (that is, cost) to future chip design is obvious.

Processor Area Becoming Memory Dominated

- On-chip SRAM contains 50–90% of total transistor count
  - Xeon: 48M/110M
  - Itanium 2: 144M/220M
- SRAM is a major source of chip static power dissipation
  - Dominant in ultra low-power applications
  - Substantial fraction in others

Slide 7.3
The die photo of Intel’s Penryn™ processor makes the importance of SRAM even clearer. The large caches are immediately visible. We have circled in red just a few of the numerous other SRAM blocks on the chip. In addition to the impact on area, the power dissipation of the memory is growing relative to that of other components on the chip. This is particularly true for the leakage component of chip power. As the SRAM must remain powered on to hold its data, the large number of transistors in on-die SRAM will constantly draw leakage power. This leakage power can dominate the standby power and active leakage power budgets in low-power applications, and become an appreciable fraction of the total dissipation in others.
Chapter Outline

- Introduction to Memory Architectures
- Power in the Cell Array
- Power for Read Access
- Power for Write Access
- New Memory Technologies

Slide 7.4
We begin our discussion of memory power optimization with an introduction to memory structures with a focus on embedded SRAM. Then we describe design time techniques for lowering power in the cell array itself, for reducing power during read accesses, and for decreasing power during write accesses. Finally, we present emerging devices that show promising results for reducing power in SRAM.

Given the limited space, an in-depth discussion on the operation of SRAM memories and the prevailing trends is unfortunately not an option. We refer the reader to specialized textbooks on the topic, such as [Itoh’07].

Slide 7.5
A two-dimensional array of SRAM bit-cells is the basic building block of large SRAM memories. The dimension of each cell array is limited by physical considerations such as the capacitance and resistance of the lines used to access cells in the array. As a result, memories larger than 64–256 K\( \text{b} \) are divided into multiple blocks, as shown on this slide. The memory address contains three fields that select the block, the column, and the row of the desired word in the memory. These address bits are decoded so that the correct block is enabled and appropriate cells in that block are selected. Other circuits drive data into the cells for a write operation or drive data from the cells onto a data bus during a read. We may treat all of these peripheral circuits (e.g., decoders, drivers, control logic) as logic and apply to them the powersaving techniques from the preceding chapters of this book. The truly unique structure in an embedded SRAM is the array of bit-cells itself. In this chapter, we will focus on power-saving approaches specifically targeted at the bit-cell array.

Slide 7.6
In standard CMOS logic, the trade-off between power and delay tends to take precedence over other metrics. Although we certainly need combinational logic to function properly, static CMOS is sufficiently robust to make functionality relatively easy to achieve (at least for the time being). In
memory, this is not necessarily the case. The need for ever more storage density makes area the dominant metric — even though power is recently gaining ground in terms of its importance. SRAMs give up some of the important properties of static CMOS logic (e.g., large noise margins, non-ratioed circuits) to reduce the cell area. A typical cell is thus less robust (closer to failure) than typical logic. As we have discussed quite a number of times, the rapid increase in process variations that has accompanied CMOS process technology scaling causes circuit parameters to vary. Although process variations certainly impact logic, they have an even more profound impact on SRAM due to the tighter margins. One of the most insidious sources of variation is random dopant fluctuation (RDF), which refers to the statistical variation of the number and position of doping ions in a MOSFET channel. RDF leads to significant variation in the threshold voltage of transistors with an identical layout. This means that physically adjacent memory cells exhibit different behaviors based on where their devices fall in the distribution of threshold voltages. As a result, important metrics related to the cell, such as delay and leakage, should be considered a distribution rather than a constant. When we further consider that embedded SRAMs may have many millions of transistors, we realize that some cells will necessarily exhibit behavior well out of the tail of the metric distribution (as far as 6σ or 7σ).

Although the power–delay trade-off certainly exists in memory, the more pressing issue in deeply scaled technologies is the trade-off between power and functional robustness (and area as a close second). Turning the circuit knobs to reduce SRAM power degrades the robustness of the array, so functionality is usually the limiting factor that prevents further power reduction. This means that, for SRAM, the primary goal when attempting to lower power is to achieve savings while maintaining correct operation across the entire array. The dominant aspects of functionality are readability, writability, data retention, and soft errors. In this chapter, we focus on the first two. We will look at data retention limits in detail in Chapter 9. Soft-error rates (SERs) for modern SRAM are increasing because each bit-cell uses less charge to store its data owing to smaller capacitance and lower voltage. As a result, the bits are more easily upset by cosmic rays and alpha particles. We will not discuss soft-error immunity in detail, although there is a variety of techniques that help to reduce the SER, such as error correction and bit interleaving.

The theme of this chapter centers around these threats to SRAM functionality: To save power in SRAM, introduce new techniques to improve robustness, and trade off that gained robustness to lower power subsequently.

**Slide 7.7**

Before we begin to reduce power in SRAM, we ought to ask ourselves, “Where does SRAM power go?” Unfortunately, this is a difficult question to answer. An expedition through the literature
Where Does SRAM Power Go?

- Numerous analytical SRAM power models
- Great variety in power breakdowns
- Different applications cause different components of power to dominate
- Hence: Depends on applications: e.g., high speed versus low power, portable

serve a huge variety of purposes. Even on the same chip, one large high-speed four-way cache may sit next to a 1 Kb rarely accessed look-up-table. From chip to chip, some applications require high-performance accesses to a cache nearly every cycle, whereas some portable applications need ultra low-energy storage with infrequent accesses. As a result, the optimal SRAM design for one application may differ substantially from that for another. As the constraints and specifications for each application determine the best SRAM for the job, we restrict ourselves to a survey of the variety of power-saving techniques that fall under the trade-off theme of this text book. Again, for SRAM, the trade-off is usually for functional robustness rather than for delay.

Slide 7.8

The circuit schematic on this slide shows the traditional topology for an SRAM bit-cell. It consists of six transistors, and is thus often called the 6T cell. The primary job of the bit-cell is to store a single bit of data, and it also must provide access to that data through read and write functions. The cell stores a single bit of data by using the positive feedback inherent in the back-to-back inverters formed by the transistors M1, M3 and M4, M6. As long as power is supplied to the cell and the wordline (WL) remains low (so that the transistors M2 and M5 are off), data at node Q will drive node QB to the opposite value, which will in turn hold the data at node Q. In this configuration, the voltages on the bitlines (BL and BL [or BLB]) do not impact the functionality of the bit-cell. To write the bit-cell (change the data in the bit-cell), we must overpower the positive feedback inside the cell to flip it to the opposite state. For example, if Q = 1 and QB = 0, we must drive Q to 0 and QB to 1 in order to write a new value into the cell. To accomplish this, we can drive the new data onto the BLs (e.g., BL = 0 and BLB = 1) and then assert the WL. This write operation is clearly ratioed, as it creates a fight between the devices inside the cell and the access transistors (M2 and M5). The NMOS access transistors are good at passing a 0, so we will rely on the side of the cell with a BL at ground to execute the write. To ensure that this works properly, we size M2 (M5) to win the fight with M3 (M6) so that we can
pull the internal node that is high down to a 0 to flip the cell. We would also like to use the same access transistors (M2 and M5) to read the contents of the bit-cell to keep the size of the cell as small as possible. This means that we should be careful to avoid driving a BL to 0 during a read operation so that we do not inadvertently write the cell. To prevent this problem, we precharge both BLs to $V_{DD}$, and then allow them to float before asserting the WL. We can thus consider the BLs to be capacitors that are charged to $V_{DD}$ at the onset of the read access. The side of the cell that stores a 0 will slowly discharge its BL — the read is slow because the cell transistors are small and the BL capacitance is relatively large — while the other BL remains near $V_{DD}$. By looking at the differential voltage that develops between the BLs, we can determine what value the cell is storing.

**Slide 7.9**

The traditional metrics of area, power, and delay apply to an SRAM. The driving metric has been area for a long time due to the large number of cells in SRAM arrays. However, power is becoming increasingly important to the point of rivaling area as the driving metric for the reasons that we described on Slide 7.3. Tuning the access delay of a memory is also of essence, but many embedded memories do not need to be super high-speed. Delay can thus be traded off to save area or power. As mentioned on Slide 7.6, robustness issues have floated to the top as a result of increasing process variations. This makes functionality a primary concern, and it limits the extent to which we can turn design knobs to lower power. A very useful metric to measure the robustness of a cell is the static noise margin (SNM), which is a measure of how well the cell can hold its data. An idle cell can generally hold its data quite well (i.e., the “hold SNM” is large), although the SNM decreases with a lowering of the supply voltage. $V_{DD}$ scaling is a good knob for reducing leakage power, but the hold SNM places an upper limit on the achievable savings using this approach. We define the data retention voltage (DRV) as the lowest voltage at which a cell (or array of cells) can continue to hold its data. We will talk more about DRV in Chapter 9.

During a read access, the SNM is degraded due to the voltage-dividing effect that occurs between the access transistor and the drive transistor on the side of the bit-cell that holds a 0. This means that the bit-cell is most susceptible to losing its data during a read access. This type of read upset also becomes more likely as the power supply of the bit-cell is decreased.

As a successful write into an SRAM cell depends upon a sizing ratio, it also becomes more likely to fail in the presence of process variations. Specifically, variations that strengthen the PMOS transistors in the cell relative to the access transistors can be detrimental. An intended write may not occur if the access transistor cannot overpower the back-to-back inverters in the cell. The following slides discuss these metrics in more detail.
Slide 7.10

This slide provides a detailed illustration of the static noise margin (SNM) of a cell. The circuit schematic shows a cell with DC-voltage noise sources inserted into the cell. For now, let us assume that the value of these sources is $V_n = 0$ V. The thick lines in the plot show the DC characteristics of the cell for the condition where there is no noise. The voltage transfer characteristic (VTC) curves cross at three points to make two lobes. The resulting graph is called the butterfly plot of the cell as the lobes resemble butterfly wings. The two crossing points at the tips of the lobes are the stable points, whereas the center crossing is a meta-stable point.

Consider now the case where the value of the noise sources $V_n$ start to increase. This causes the VTC of inverter 2 to move to the right, and the VTC of inverter 1 moves downward. The cell remains bistable (i.e., it holds its data) as long as the butterfly plot keeps its two lobes. Once the VTCs have moved so far that they only touch in two locations, one lobe disappears and any further increases in $V_n$ result in a monostable bit-cell that has lost its data. This value of $V_n$ is the static noise margin. The thin lines on the plot illustrate the VTCs in this condition. They touch at the corner of the largest square that can be inscribed inside the lobe of the original butterfly plot. The SNM is now defined as the length of the side of the largest square inside the butterfly plot lobe. If the cell is imbalanced (e.g., due to transistor sizing or process variations) – one lobe is smaller than the other in that case – then the SNM is the length of the side of the largest square that fits inside the smallest of the two lobes. This indicates that the bit-cell is more susceptible to losing one particular data value.

Slide 7.11

Process scaling causes the SNM of SRAM bit-cells to degrade. This slide shows simulations from predictive technology models (PTMs) of the SNM in 65 nm, 45 nm, and 32 nm. The upper plot shows that the typical SNM degrades with technology scaling and with voltage reduction. This means that it is harder to make a robust array in a scaled technology, and that lowering supply voltage to reduce power degrades the cell stability. Furthermore, this plot confirms that the read SNM is quite a bit smaller than the hold SNM.

If this story is not already bad enough, variations make it substantially worse. The bottom plots show distributions of the read SNM for the different technology nodes. Clearly, the tails of these distributions correspond to cells with vanishingly small noise margin, indicating that those cells will be quite unstable during a read access even in traditionally safe SRAM architectures. For the 32 nm technology, a substantial number of cells exhibit an SNM at (or below) 0, indicating a read upset even in the absence of other noise sources. This degradation of stability means that SRAM
Static Noise Margin with Scaling

- Typical cell SNM deteriorates with scaling
- Variations lead to failure from insufficient SNM

Variations worsen tail of SNM distribution

Tech and $V_{DD}$ scaling lower SNM

(Results obtained from simulations with Predictive Technology Models – [Ref: PTM, Y. Cao '00])

Variability: Write Margin

In CMOS sequential logic, the most-often-used latches simplify the write process by disabling the feedback loop between the cross-coupled inverters with the aid of a switch. The SRAM cell trades off this robust write method for area. The write operation now transgresses into a ratioed fight between the write driver and one of the inverters inside the cell. This “battle” is illustrated graphically on this slide. Write drivers assert the new data values onto the bitlines through a pass gate (not shown), then the WL goes high. This connects the internal nodes of the bit-cell with the driven bitlines, and a fight ensues between the cell inverters and the driver through the access transistor. As the NMOS access transistors pass a strong 0, the BL with a 0 is well-positioned to win its fight so long as the access transistor can overpower the PMOS pull-up transistor to pull down the internal node far enough to flip the cell. We can analyze the robustness of the cell’s writability by looking at the equivalent of the butterfly plot during a write access. The bottom left-hand plot on this slide shows the butterfly plot of a bit-cell holding its data. For a successful write, the access transistors must drive the cell to a monostable...
condition. The lower right-hand plot shows a butterfly plot that no longer looks like a butterfly plot because it has successfully been made monostable (writing a 1 to $Q$). This corresponds to a negative SNM. In the upper right plot, the butterfly curve maintains bistability as indicated by the fact that both lobes of the butterfly plot persist during the write. This means that the write attempt has failed.

![Variability: Cell Writability](image)

**Slide 7.13**
As with SNM, the write margin degrades in the presence of process variations. This is illustrated in this graph, which plots the onset of negative SNM for a write operation at the different global corners of a 65 nm process (e.g., typical NMOS, typical PMOS [TT]; weak NMOS, strong PMOS [WS]). Even before we account for local variations, 600 mV is the lowest voltage at which a 6T bitcell allows for a successful write operation across global PVT (process, voltage, temperature) corners in this process. Local variations make the minimum operating voltage even higher. This indicates that successful write operation is compromised even for traditional 6T bit-cells and architectures in scaled technologies. As with read SNM, this limits the amount of flexibility that we have for trading off to save power. Hence approaches that improve functional robustness must be introduced. Only after this is accomplished can we start trading off robustness for power reduction.

**Slide 7.14**
Now that we have reviewed the traditional SRAM architecture, the bit-cell, and its important metrics, we take a look at the power consumption inside the bit-cell array when it is not being accessed. We assume for the moment that the array is still active in the sense that read or write accesses are imminent. In Chapter 9, we look at the standby case where no accesses are anticipated. Since the non-accessed array is merely holding its data, it does not consume switching power. Its power consumption is almost entirely leakage power.

Inactive cells leak current so long as the array is powered. This slide shows the primary paths for sub-threshold leakage inside the bit-cell. As we can assume that both the bitlines are precharged when the cell array is not being accessed, both BL and BLB are at $V_{DD}$. This means that the drain-to-source voltage across the access transistor on the “0”-side of the cell equals $V_{DD}$, causing that device to leak. Similar leakage occurs in the PMOS transistor on the same side of the cell, and the NMOS drive transistor on the opposite side. These three compose the dominant leakage paths inside the bit-cell.
Other leakage mechanisms are also significant in modern technologies (see earlier chapters). Most notably, leakage through the gate terminal occurs for thin-gate-oxide transistors with a large $V_{GD}$ or $V_{GS}$. Most scaled technologies have kept gate leakage to acceptable levels by slowing down the scaling of the gate oxide thickness. Emerging technologies at the 45 nm process node promise to include high-k dielectric material for the gate insulator, enabling further dielectric scaling. Assuming that this occurs, we postulate that sub-threshold leakage will continue to be the dominant leakage source in a CMOS cell. If not, the impact of gate leakage should be included in the design optimization process.

In the following slides, we examine two knobs for reducing the leakage power of an array: the threshold voltage and the peripheral voltages. These knobs can be set at design time such that SRAM leakage during active operation (e.g., when the array is ready to be accessed) decreases.

### Slide 7.15

As established earlier, the sub-threshold leakage current equals $I_{SUB} = I_0 \exp \left( \frac{V_{GS} - V_{TH} + 2aV_{DD}}{nkT/q} \right)$. We can directly observe from this equation that the threshold voltage, $V_{TH}$, is a powerful knob, exponentially reducing the off-current of a MOSFET. Hence, one SRAM leakage reduction technique is to select a technology with a sufficiently high threshold voltage. The plot on this slide shows the leakage current for a 1 Mb array (log scale) versus the threshold voltage at different temperatures. Clearly, selecting a larger $V_{TH}$ has an immediate and powerful effect on reducing the leakage current. If we assume that a high-speed memory application can tolerate 10 μA of leakage...
current at 50°C, then this plot indicates that the $V_{TH}$ of that array must be 490 mV. Likewise, the plot shows that a low-power array (0.1 μA of leakage current at 75°C) needs a $V_{TH}$ of over 710 mV if all other design parameters remain the same. This analysis not only indicates that threshold voltage can be used to control the leakage in an array, but also that it must remain fairly large if it is the lone knob used for controlling leakage power. On the negative side, the higher $V_{TH}$ decreases the drive current of the bit-cells and limits the speed of the memory.

**Slide 7.16**

One alternative to using high-$V_{TH}$ transistors for the entire cell is to selectively replace some high-$V_{TH}$ devices with low-threshold transistors. Out of the large number of possible arrangements using transistors with two threshold voltages, only a few make sense. The best choice depends upon the desired behavior of the memory and the technology at hand.

A potential shortcoming of the multiple-threshold approach is that design rules may be imposed that force FETs with different $V_{TH}$s to be further apart. If area increases can be avoided or fall within an acceptable range, dual-$V_{TH}$ cells can offer some nice advantages. One example is the cell shown on the left side of this slide (with low-$V_{TH}$ devices shaded). The cross-coupled inverters in the cell are high-threshold, thereby effectively eliminating the leakage paths through the inverters. The access transistors are low-$V_{TH}$, along with the peripheral circuits. This translates into an improved drive current during read, minimizing the read-delay degradation due to the use of high-$V_{TH}$ transistors.

The cell on the right side of the slide exploits the property that in many applications the majority of the cells in a memory store a "0". Selectively reducing the leakage of these cells hence makes sense. In fact, the leakage in a "0" cell is reduced by as much as 70% in this circuit. This obviously translates into a higher leakage for the "1" cells, but as these are a minority, the overall memory leakage is substantially reduced.

**Slide 7.17**

The use of multiple voltages provides another strong leakage reduction knob. Careful selection of the voltages inside and around the cell can decrease leakage in key devices by, for example, producing negative gate-to-source voltages. The sub-threshold current equation shows us that a negative $V_{GS}$ has the same exponential impact on leakage current as raising $V_{TH}$. In the cell shown on this slide, the WL is at 0, but the source voltage of the cross-coupled inverters is increased to 0.5 V. This sets the $V_{GS}$ for the access transistors to −0.5 V and −1.0 V for the sides of the cell holding a logical "0" and "1", respectively, producing a dramatic decrease in sub-threshold leakage. The supply voltage inside the cell must consequently be increased to maintain an adequate
Multiple Voltages

- Selective usage of multiple voltages in cell array
  - e.g., 16 fA/cell at 25°C in 0.13 μm technology

- High $V_{TH}$ to lower sub-$V_{TH}$ leakage
- Raised source, raised $V_{DD}$, and lower BL reduce gate stress while maintaining SNM

[Ref: K. Osada, JSSC’03]

SNM. In this particular implementation, the author uses high-$V_{TH}$ FETs in conjunction with voltage assignment to achieve a 16 fA/cell leakage current in a 130 nm technology.

In summary, increasing the threshold voltage is a strong knob for lowering sub-threshold leakage in the “non-active” cell, but higher-threshold devices translate into longer read or write latencies. Lowering the cell voltage or introducing multiple voltages also helps to reduce leakage power, but must be executed with care to avoid a degradation of the cell robustness. Be aware also of the hidden cost and area penalties associated with some of these techniques. For instance, extra thresholds mean extra masks – fortunately, most state-of-the-art processes already offer two thresholds. Providing extra supply voltages imposes a system cost due to the extra DC–DC converter(s) required, whereas routing multiple supply voltages incurs cell and periphery area overhead. All of these sources of overhead must be weighed against the power savings for a complete design.

In the next set of slides, we concentrate on how to impact power during a read access.

Power Breakdown During Read

- Accessing correct cell
  Decoders, WL drivers
  - For Lower Power:
    - hierarchical WLs
    - pulsed decoders

- Performing read
  - Charge and discharge large BL capacitance
  - For Lower Power:
    - SAs and low BL swing
    - Hierarchical BLs
    - Lower BL precharge
    - Lower $V_{DD}$
      - May require read assist

Slide 7.18
As the read access involves active transitions in the SRAM, the dominant source of power consumption during a read is switching power. This slide provides a general conceptual breakdown for where the switching power is dissipated during a read access. When the address is first applied to the memory, this address is decoded to assert the proper wordline. The decoded signals are buffered to drive the large capacitance of the wordline. The decoder and wordline drivers are nothing else but combinational logic, and techniques to manage power in this style of networks were treated in-depth in previous chapters.
We do not revisit these circuit-level techniques here, but restrict ourselves to techniques that specifically make sense in the memory context.

Once the decoded wordline signal reaches the cell array, the selected bit-cell selectively discharges one of the (precharged) bitlines. Owing to the large bitline capacitance, recharging it during the next precharge phase consumes a significant amount of power. Clearly, reducing the amount of discharge helps to minimize dissipation. It is traditional in SRAMs to use a differential sense amplifier (SA) to detect a small differential signal on the bitlines. This not only minimizes the read latency but also allows us to start precharging the bitline after it has only discharged a small fraction of $V_{DD}$, thus reducing power as well. On top of this, there are several other approaches for lowering power during this phase of the read access, some of which are described in the following slides.

![Hierarchical Wordline Architecture](image)

- Reduces amount of switched capacitance
- Saves power and lowers delay

[Refs: Rabaey, Prentice'03, T. Hirose, JSSC'90]

Slide 7.19
The capacitance of a wordline in an SRAM array can be quite large. It consists of the gate capacitance of two access transistors per bit-cell along the row in the array plus the interconnect capacitance of the wires. This capacitance gets even larger if a single wordline is deployed for accessing rows across multiple blocks in a large SRAM macro. To counter this, most large memories use a hierarchical wordline structure similar to the one shown on this slide. In this structure, the address is divided up into multiple fields to specify the block, block group, and column, for example. The column address is decoded into global wordlines, which are combined with select signals to produce sub-global wordlines. These in turn are gated with the block-select signals to produce the local wordlines. Each local wordline can thus be shorter and have less capacitance. This hierarchical scheme saves power and lowers delay by reducing the amount of capacitance that is switched on the wordline. The approach also allows for additional power savings by preventing wordlines in non-accessed blocks from being activated, which would cause dummy read operations in those blocks.

Slide 7.20
Dividing large lines into a hierarchy of smaller lines works for bitlines just the same way as it works for wordlines. The bitlines typically do not discharge all the way because of the sense amplifiers. Nevertheless, the large capacitance of these lines makes discharging them costly in terms of power and delay. Decreasing the number of cells on a local bitline pair reduces the delay and power consumption of the read access. The local bitlines can be recombined into global bitlines that provide the final data value from the read. The emergence of bitline leakage as a
Hierarchical Bitlines

- Divide up bitlines hierarchically
  - Many variants possible
- Reduces $RC$ delay, also decreases $CV^2$ power
- Lower BL leakage seen by accessed cell

BL Leakage During Read Access

- Leakage into non-accessed cells
  - Raises power and delay
  - Affects BL differential

Slide 7.21

Bitline leakage refers to leakage current paths that flow from the bitlines into the bit-cells along an SRAM column. We have already identified this leakage path from a more local perspective on Slide 7.14. Bitline leakage is actually more problematic than described there, as it degrades the ability of the SRAM to read properly. This is illustrated in this slide, where a single cell tries to drive a “1” on the line while other cells on the column hold a “0”. For this data vector, all of the non-accessed cells contribute leakage currents that oppose the (very small) drive current from the accessed bit-cell. As a result, the bitline, which should keep its precharged voltage at $V_{DD}$, may actually discharge to some lower voltage. As a consequence, the difference in its voltage from the voltage on the opposite bitline (not shown), which is supposed to discharge, is diminished. At the very least, this leakage increases the time for the sense amplifier to make its decision (hence raising the read latency).

The impact of the data distribution on the memory-access time is shown in the graph on the left. Clearly, if all of the non-accessed cells contain data opposite to that of the accessed cell, the delay increases dramatically. The variation of the access time also increases. In the worst case, bitline
leakage into a large number of non-accessed cells can potentially become equal or larger than the drive current of the accessed cell, leading to a failure of the SRAM to read the cell. The number of bit-cells along a bitline pair must be carefully selected to prevent this type of bitline-leakage induced error.

### Slide 7.22
A number of possible solutions to combat bitline leakage are listed on this slide.

Hierarchical bitlines reduce the number of cells connected to a line, but increase complexity and area by requiring more peripheral circuits per bit-cell. Raising the virtual ground node inside the non-accessed bit-cells lowers leakage from the bitline at the cost of added area and reduced SNM. Reducing the wordline voltage below zero (negative WL) exponentially decreases the sub-threshold leakage through the access transistors, but this approach may be limited by gate current, which increases as a result of the large $V_{DG}$. Lengthening the access transistors lowers leakage at the cost of a decreasing drive current. Alternative bit-cells have been proposed, such as an 8-transistor (8T) cell that uses two extra access transistors (that are always off) to couple the same amount of leakage current to both bitlines. This cell successfully equalizes the leakage on both bitlines, but it does so by making the leakage worst-case. Hence, it is only successful in reducing the impact of bitline leakage on delay, not on power. Some active-compensation approaches have been proposed that measure the leakage on the bitline and then apply additional current to prevent the erroneous discharging of the high bitline. These sorts of schemes increase complexity and tend to focus on decreasing the delay at the expense of power. Reducing the precharge voltage is another approach (as it merits further discussion, we postpone its description to the next slide).

All of these techniques can help with the bitline leakage problem, but translate into some sort of trade-off. The best solution for a given application, as always, depends upon the specific circumstances and settings.

The bar graph on this slide provides a high-level comparison of the effectiveness of several of these techniques (obtained by simulations using the predictive modeling technology or PTM) with respect to read-access latency. The conventional approach and the 8T cell do not work all the way to the 32 nm node. Raising the virtual ground in non-accessed cells, using a negative wordline voltage, and subdividing the array with hierarchical bitlines all help to make the array less sensitive to bitline leakage. Again, the trade-offs that each approach makes to accomplish this must be carefully weighed, and included in the exploration process.
Lower Precharge Voltage

Lower BL precharge voltage decreases power and improves Read SNM
- Internal bit-cell node rises less
- Sharp limit due to accidental cell writing if access FET pulls internal ‘1’ low

Slide 7.23
Reducing the precharge voltage on the bitlines below the traditional $V_{DD}$ value helps to reduce bitline leakage into the non-accessed bit-cells, because of the lower $V_{DS}$ across the access transistors. Since the access transistor that drives a 1 onto the bitline during a read does not turn on unless the bitline drops to $V_{DD} - V_{TH}$ anyway, the lower precharge voltage does not negatively affect the read itself. In fact, by weakening the access transistor on the 0 side of the cell, the lower precharge voltage actually makes the read more robust by improving the read SNM. The chart shows that a lower precharge value can improve the read SNM by over 10%, in conjunction with a lower leakage power. One simple way to implement this method is to precharge using NMOS devices instead of the traditional PMOS. The chart also indicates the major limitation to this approach: if the precharged bitline is at too low a voltage, the cell may be inadvertently written during a read access. This is indicated by a sharp roll-off in the read SNM.

Slide 7.24
A discussion of the power saving-approaches during read is incomplete without a closer look at “classic” $V_{DD}$ scaling. Lowering the supply voltage of an SRAM array during active mode clearly decreases the switching power consumed by that array ($P = fC V^2$). It also decreases leakage power as $P = V_{DD}^2 I_{off}$, and $I_{off}$ mostly decreases as a result of the DIBL effect. This double power-wins comes at the cost of increased access delays. We also know by now that the reduction in the operational $V_{DD}$ is quite limited owing to functional barriers such as SNM and read/write margins.

There are two solutions to this problem. The first is to admit defeat so far as the array is concerned by using high-$V_{TH}$ devices and maintaining a high $V_{DD}$ to provide sufficient operating margins and speed. The peripheral circuits, on the other hand, can be scaled using traditional voltage scaling as they are essentially combinational logic. The complication to this approach is the
need for level conversion at the interface between periphery and the array. The second solution is to recover the lost margin (read margin, as we are talking about the read access here) using read-assist techniques. These are circuit-level approaches that improve the read margin, which in turn can be used to reduce the $V_{DD}$. Examples of read-assist approaches include lowering the BL precharge voltage, boosting the bit-cell $V_{DD}$, pulsing the WL briefly, re-writing data to the cells after a read, and lowering the WL voltage. All of these approaches essentially work to sidestep the read-upset problem or to strengthen the drive transistor relative to the access transistor so as to reduce read SNM. The slide provides a number of references for the interested reader.

**Slide 7.25**

We now move on to look at the power consumed during the write access. We can partition the power consumed during a write access as belonging to two phases, similar to the way that we partitioned the read access. First, we must access the proper cells in the SRAM array, and second we must perform the write. The cell access is basically the same as for the read access. Once the correct local wordline is asserted, the new data must be driven into the accessed bit-cell to update the cell to the new value. The traditional mechanism for accomplishing this is to drive the differential value of the new data onto the bitlines in a full-swing fashion. As a subsequent write with a different data or a subsequent read (with precharge) will charge up the discharged bitline, this approach can be costly in terms of power. In fact, the power for a write access is typically larger than that for a read access owing to this full-swing driving of the bitlines. Fortunately write operations tend to occur less commonly than read operations. We examine techniques that use charge sharing, exploit data dependencies, and use low-swing bitlines to reduce the power consumption associated with the write access in the following three slides.

**Slide 7.26**

The full swing on the bitlines during a write operation seems particularly wasteful if successive writes are performed in the same block. In this case, the bitlines are charged and discharged according to the incoming data. The large capacitance of the bitlines causes significant $CV^2$ power consumption. If consecutive writes have different data values, then one bitline must discharge while the opposite bitline charges up for the next write. Instead of performing these operations separately, we can apply charge recycling to reduce the power consumption. This slide shows a simple example of how this works. The key concept is to introduce a phase of charge-sharing in between phases of driving data. Assume that the old values are 0 and $V_{DD}$ on BL and BLB, respectively. During the charge-sharing phase, the bitlines are floating (e.g., not driven) and shorted together. If they have the same capacitance, then they will each settle at $V_{DD}/2$. Finally, the bitlines are driven to their new values. As BL only needs to be charged to $V_{DD}$ from $V_{DD}/2$, the
Charge recycling to reduce write power

- Share charge between BLs or pairs of BLs
- Saves for consecutive write operations
- Need to assess overhead

Basic charge recycling – saves 50% power in theory

Slide 7.27
A different type of approach to reducing write power is based on the earlier observation that one of the data values is more common. Specifically, for the SPEC2000 benchmarks, 90% of the bits in the data are 0, and 85% of the bits in the instruction memory are 0 [Chang’04]. We can take advantage of the predominance of 0s in a few ways.

First, we can use a write methodology that presets the BLs prior to each write based on the assumption that all of the bits will be 0. Then, as long as a word contains more 0s than 1s, the power consumed for driving the BLs to the proper values is reduced compared to the case in which both BLs are precharged to \( V_{DD} \). In addition, words with more 1s than 0s can be inverted (keeping track of this requires one extra bit per word) to conform to the precharge expectation. This approach can reduce write power by up to 50% [Chang’99].

Second, an alternative bit-cell introduces asymmetry to make the power consumed when writing a 0 very low. As this is the common case, at least for some applications, the average write access power can be reduced by over 60% at a cost of 9% area increase. These approaches point out the intriguing concept that an application-level observation (i.e., the preponderance of 0s) can be
exploited at the circuit level to save power. Of course, this is in accord with our discussion on Slide 7.7 regarding the close relationship between the intended application for a specific SRAM design and the memory design trade-offs.

**Slide 7.28**
As the large bitline swing for a traditional write is the primary source of power dissipation, one seemingly obvious approach to is to reduce the swing on the bitlines. Doing so, of course, makes the access transistors less capable of driving the new data into the cell. This slide illustrates a solution that utilizes low-swing bitlines for writing along with an amplification mechanism in the cell to ensure successful write. The idea requires that a power gating NMOS footer be placed in series with \( V_{\text{SS}} \) for the bit-cell. This device (driven by SLC in the schematic) can be shared among multiple bits in the word. Prior to the onset of the write, this footer switch is disabled to turn off the NMOS driver FETs inside the bit-cell. The WL goes high, and the internal nodes of the bit-cell are set high. The (weakened) access transistors are able to do so, as the pull-down paths in the cell are cut off. Then, the bitlines are driven to \( V_{\text{DD}}-V_{\text{TH}} \) and to \( V_{\text{DD}}-V_{\text{TH}}-\Delta V_{\text{BL}} \), respectively, according to the input data. This bitline differential is driven into the bit-cell, and it is subsequently amplified to full swing inside the bit-cell after WL goes low and SLC goes high. This scheme can save up to 90% of the write power [Kanda’04].

**Slide 7.29**
As with read-power reduction techniques, the fundamental limit to most power-reducing techniques is the reduced functional robustness (i.e., the write margin becomes too small, and some cells become non-writable). Again the approach for pushing past this hurdle is to
improve the write margin using some circuit innovations, and to trade off the improved robustness for power savings.

On this slide, we refer to a few of the many successful mechanisms for enabling this trade-off. Raising the voltage of the wordline during a write access relative to the $V_{DD}$ does strengthen the access transistor relative to the cell pull-up transistors, creating a larger write margin and allowing for lower-voltage operation. Collapsing the $V_{DD}$ or raising the $V_{SS}$ inside of the bit-cell has the equivalent effect of reducing the strength of the cell relative to the access transistors. Finally, we have already described a method that provides amplification inside the cell. The references can help the interested reader to explore further.

Slide 7.30
Most of the techniques described up to this point use the basic 6T as the basis. A more dramatic approach is to explore alternatives to the 6T bit-cell itself. These alternative bit-cells usually improve on the 6T cell in one or more ways at the expense of a larger area. A number of cells that may replace the 6T cell in some usage scenarios are proposed in the following slides. Even more dramatic (and much needed) changes to the SRAM cell could come from modifying the CMOS devices themselves (or even abandoning CMOS altogether). A number of new devices that offer enticing properties and may potentially change the way we design memories are therefore discussed as well. There is a huge amount of very creative activity going on in this field, and it will be no surprise to the authors if this leads one day to a very different approach of implementing embedded memory.

As we have repeatedly described, the key obstacle to power savings in SRAM is degraded functional robustness. Non-traditional bit-cells can provide improved robustness over the 6T bit-cell, which we can then trade off for power savings. In general, this leads to a larger area owing to the additional transistors.

One attractive alternative to the 6T with a number of interesting properties is the 8T, as shown on this slide. A 2T read buffer is added to the 6T cell. This extra read buffer isolates the storage node during a (single-ended) read so that the read SNM is no longer degraded. By decoupling the drive transistor from the storage node, this cell also allows for larger drive current and shorter read access times. In addition, the extra read buffer effectively enables separate read and write ports. This can improve the access rate to a memory by overlapping writes and reads. These improvements in read robustness allow the 8T to operate at lower supply voltages, and it does so without using extra voltage supplies.
Of course, these improvements come at a cost. The most obvious penalty is extra area, although a tight layout pattern keeps the array overhead down. Furthermore, the extra robustness of the cell may allow for the clustering of more cells along a single column, reducing the amount of peripheral circuitry required. The area overhead for the entire SRAM macro thus is less than the overhead in a single cell. The main challenge in using this cell is that it imposes architectural changes (i.e., two ports), which prevent it from acting as a direct replacement for 6T without needing a major macro redesign. However, the 8T cell is a wonderful example of how non-traditional bit-cells may offer improvements in robustness that can be exploited to improve power efficiency.

### Slide 7.31

The 8T transistor statically isolates the storage node from the bitlines during a read operation. The two alternatives presented in this slide achieve the same effect using a pseudo-static approach. Both cells operate on the same principle, but the left-hand cell provides a differential read, whereas the right-hand cell uses a single-ended read. When the cells hold data, the extra wordlines (WLW, WLb) remain high so that the cell behaves like a 6T cell. During the read access, the extra wordline is dropped (WLW = 0, WLb = 0). This isolates the storage node, which holds its data dynamically while the upper part of the cell discharges the proper bitline. As long as the read access is sufficiently short to prevent the stored data from leaking away, the data is preserved. These cells each add complexity to the read operation by requiring new sensing strategies on the bitline(s).

### Slide 7.32

A different tactic for reducing embedded SRAM power is to replace standard CMOS transistors with alternative devices. A slew of CMOS-replacement technologies are under investigation in labs around the world, and they range from minor modifications to CMOS, all the way to completely unrelated devices. Out of the many options, we take a brief look at one structure that is compatible with CMOS technologies, and which many people project as a likely direction for CMOS.

This slide shows a FINFET transistor (see also Chapter 2) that uses a vertical fin of silicon to replace the traditional planar MOSFET. Two types of devices can be constructed along this basic concept. The double-gated (DG) MOSFET is a vertically oriented MOSFET with a gate that wraps around three sides of the MOS channel. This allows the gate terminal to retain better control over the channel. In the back-gated (BG) MOSFET, the top part of the gate is etched away to leave
Emerging Devices: Double-gate MOSFET

- Emerging devices allow new SRAM structures
- Back-gate biasing of thin-body MOSFET provides improved control of short-channel effects, and re-instates effective dynamic control of $V_{TH}$.

Double-gated (DG) MOSFET
- Independent front and back gates
- One switching gate and $V_{TH}$ control gate

Back-gated (BG) MOSFET
- Gate length $L_g$
- Gate height $H_{gate} = W/L$
- Independent front and back gates
- One switching gate and $V_{TH}$ control gate

[Ref: Z. Guo, ISLPED’05]

6T SRAM Cell with Feedback

- Double-Gated (DG) NMOS pull-down and PMOS load devices
- Back-Gated (BG) NMOS access devices dynamically increase $I_i$-ratio
  - SNM during read ~300 mV
  - Area penalty ~18%

Slide 7.33

Using these two devices, we can re-engineer the 6T SRAM cell, so that butterfly diagrams as shown in this slide are obtained. The SNM for the DG-MOS bit-cell is quite similar to that of a traditional CMOS cell; the read SNM is degraded owing to the voltage-dividing effect between the access transistor and the drive transistor.

This can be remedied by connecting back-gate terminals of the BG-MOS access transistors as indicated by the red lines in the schematic, so that feedback is provided during a read access. When the storage node is high or low, the $V_{TH}$ of the access transistor is raised or lowered, respectively. In the latter case, the access transistor becomes stronger, effectively increasing the $I_i$-ratio of the cell. The bottom-right butterfly plot shows that this feedback results in a significantly improved read SNM for the cell that uses the BG-MOS devices.

This example demonstrates that device innovations can play a big role in the roadmap for embedded memory in the years to come. However, as always, the creation of a new device is
only the first step in a long chain of events that ultimately may lead to a manufactureable technology.

**Summary and Perspectives**

- **Functionality is main constraint in SRAM**
  - Variation makes the outlying cells limiters
  - Look at hold, read, write modes
- **Use various methods to improve robustness, then trade off for power savings**
  - Cell voltages, thresholds
  - Novel bit-cells
  - Emerging devices
- **Embedded memory major threat to continued technology scaling – innovative solutions necessary**

As we have seen repeatedly, process scaling and variations challenge the functionality of modern embedded SRAMs. The large sizes of embedded SRAM arrays, along with local variations, require us to examine the far tails (>6σ) of distributions to identify cells that will limit the array’s functionality. Depending upon the application and the operating environment, the limiting conditions can occur during hold, read, or write operations. As robustness is so critical, the most effective method for saving power is to apply techniques to the memory that improve functional robustness. The resulting surplus of functional headroom can then be traded off for power savings. A number of techniques for doing so using device threshold voltages, cell and peripheral supply voltages, novel cells, and emerging devices have been presented.

In the long term, only novel storage devices can help to address the joined problem of power dissipation and reliability in memory. While waiting for these technologies to reach maturity (which may take some substantial amount of time and tax your patience), it is clear that in the shorter term the only solution is to take an area-penalty hit. Another option is to move large SRAM memories to a die different from that of the logic, and to operate it on larger supply and threshold voltages. Three-dimensional packaging techniques can then be used to reconnect logic and memory.
References (cont.)


References (cont.)

that is, the standby mode. In an ideal world, this would mean that the \textit{dynamic power consumption should be zero} or very small. Moreover (given the constant ratio), \textit{static power dissipation should be eliminated as well}. Although the former can be achieved through careful management, the latter is becoming harder with advanced technology scaling. When all transistors are leaky, completely turning off a module is hard. In this chapter, we discuss a number of circuit and system techniques to keep both dynamic and static power in standby to an absolute minimum. As standby power is the main concern in memories (and as memories are somewhat special anyhow), we have relegated the discussion on them to Chapter 9.
Chapter Outline

- Why Sleep Mode Management?
- Dynamic power in standby
  - Clock gating
- Static power in standby
  - Transistor sizing
  - Power gating
  - Body biasing
  - Supply voltage ramping

Slide 8.2
We start the chapter with a discussion on the growing importance of reducing standby power. Next, we analyze what it takes to reduce dynamic power in standby to an absolute minimum. The bulk of the chapter is devoted to the main challenge: that is, the elimination (or at least, minimization) of leakage during standby. Finally, some future perspectives are offered.

Slide 8.3
With the advent of mobile applications, the importance of standby modes has become more pronounced, as it was realized that standby operation consumes a large part of the overall energy budget. In fact, a majority of applications tend to perform in a bursty fashion – that is, they exhibit short periods of intense activity interspersed with long intervals of no or not much activity. This is the case even in more traditional product lines such as microprocessors. Common sense dictates that modules or processors not performing any task should consume zero dynamic and also (preferably) zero static power.
When power became an issue, this problem was quickly corrected as shown in the traces for the Pentium-2.

**Slide 8.4**

This was not a common understanding. In the long-gone days when power in CMOS designs did not rate very highly on the importance scale, designers paid scant attention to the power dissipation in unused modules. One of the (by now) classic examples of this neglect is the first Intel Pentium design, for which the power dissipation peaked when the processor was doing the least—that is, executing a sequence of NOPs.

**Slide 8.5**

The main source of dynamic energy consumption in standby mode is the clock. Keeping the clock connected to the flip-flops of an idle module not only adds to the clock loading, but may cause spurious activity in the logic. In fact, as the data that is applied under those conditions is actually quite random, activity may be maximized as we have discussed earlier. This wasteful bit-flipping is avoided by two design interventions:

- Disconnect the clock from the flip-flops in the idle module through clock gating.
- Ensure that the inputs to the idle logic are kept stable. Even without a clock, changes at the inputs of a combinational block cause activity.

Clock gating a complete module (rather than a set of gates) makes the task a lot easier. However, deciding whether a module, or a collection of modules, is idle may not always be straightforward. Though sometimes it is quite obvious from the register-transfer level (RTL) code, normally it requires an understanding of the operational system modes. Also, clock gating can be more effective if modules that are idle simultaneously are grouped. What this basically says is that standby-power management plays at all levels of the design hierarchy.
logic determines which data path units are needed for its execution, and subsequently set their Enable signals to 1.

As the inputs of the logic module are connected to the register file, they remain stable as long as the clock is disabled. In the case that the inputs are directly connected to a shared bus, extra gates must be inserted to isolate the logic.

Observe that the gated clock signal suffers an additional gate delay, and hence increases the skew. Depending upon the time in the design process it is inserted, we must ensure that this extra delay does not upset any critical set-up and hold-time constraints.

**Slide 8.6**

One possible way of implementing clock gating is shown in this slide. The clock to the register files at the inputs of an unused module is turned on or off using an extra AND gate controlled by an Enable signal. This signal is either introduced explicitly by the system- or RTL-designer, or generated automatically by the clock synthesis tools. Take for instance the case of a simple microprocessor. Given an instruction loaded in the instruction register (IR), the decoding

**Slide 8.7**

There is no doubt that clock gating is a truly effective means of reducing standby dynamic power. This is illustrated numerically with the example of an MPEG4 decoder [Ohashi'02]. Gating 90% of the flip-flops results in a straight 70% standby power reduction. This clearly indicates that there is NO excuse for not using clock gating in today’s power-constrained designs.
Clock Gating

- Challenges to skew management and clock distribution (load on clock network varies dynamically)
- Fortunately state-of-the-art design tools are starting to do a better job
  - For example, physically aware clock gating inserts gates in clock tree based on timing constraints and physical layout

Power savings  Simpler skew management, less area

keep the gaters close to the registers. This allows for a fine-grain control on what to turn off and when. It comes at the expense of a more complex skew control and extra area. Another option is to move the gating devices higher up in the tree, which has the added advantage that the clock distribution network of the sub-tree is turned off as well – leading to some potentially large power savings. This comes at the expense of a coarser control granularity, which means that modules cannot be turned off as often.

Given the complexity of this task, it is fortunate that state-of-the-art clock synthesis tools have become more adept in managing the skew in the presence of clock gating. This will be discussed in more detail later, in the chapter on design methodology for power (Chapter 12).

Clock Hierarchy and Clock Gating

Example: Clock distribution of dual-core Intel Montecito processor

“Gaters” provided at lower clock-tree levels
Automatic skew compensation

© IEEE 2005

Slide 8.8
Yet, as mentioned, these gains do not come for free, and present an extra burden on the designers of the clock distribution network. In addition to the extra delay of the gating devices, clock gating causes the load on the clock network to vary dynamically, which introduces another source of clock noise into the system.

Let us, for instance, explore some different options on where to introduce the gating devices in the clock-tree hierarchy. One possible solution is to...
the central clock to the frequency expected for the different clock zones. The downstream clock network employs both active deskew (in the second-level clock buffers or SLCBs, and in the regional active deskew or RAD) and fixed deskew, tuned via scan (using the clock vernier devices or CVDs). The latter allow for final fine-tuning. Gaters provide the final stage of the network, enabling power saving and pulse shaping. A total of 7536 of those are distributed throughout the chip. Clock gating clearly has not simplified the job of the high-performance designer!

Slide 8.10
The introduction of clock gating succeeds in virtually eliminating the dynamic power dissipation of the computational modules during standby. However, although the ends of the clock tree have been disconnected, the root is still active and continues to consume power. Further power reductions would require that the complete clock distribution network and even the clock generator (which typically includes a crystal-driven oscillator and a phase-locked loop) are put to sleep. Although the latter can be turned off quite quickly, bringing them back into operation takes a considerable amount of time, and hence only makes sense if the standby mode is expected to last for considerable time.

Many processors and SoCs hence feature a variety of standby (or sleep) modes, with the state of the clock network as the main differentiator. Options are:

- Just clock gating
- Disabling the clock distribution network
- Turning off the clock driver (and the phase-locked loop)
- Turning off the clock completely.

In the latter case, only a wake-up circuit is kept alive, and the standby power drops to the microwatt range. Companies use different names for the various modes, with sleep mode typically reserved for the mode where the clock driver is turned off. It may take tens of clock-cycles to bring a processor back to operation from sleep mode.

Slide 8.11
The choice of the standby modes can be an important differentiator, as shown in this slide for a number of early-day low-power microprocessors. The Motorola PowerPC 603 supported four different operation modes, ranging from active, to doze (clocks still running to most units), nap (clock only to a timer unit), and sleep (clock completely shut off). The MIPS on the other hand did not support a full sleep mode, leading to substantially larger power dissipation in standby mode.

The MSP430™ microcontroller from Texas Instruments shows the state-of-the-art of standby management. Using multiple on-chip clock generators, the processor (which is actively used in
Sleep Modes in \( \mu \)Processors and \( \mu \)Controllers

[Ref: S. Gary, Springer'd95]

TI MSP430™
From standby to active in 1 \( \mu \)s using dual clock system

[Ref: T/06]

low-duty-cycle power-sensitive control applications) can go from standby (1 \( \mu \)A) to active mode (250 \( \mu \)A) in 1\( \mu \)s. This rapid turnaround helps to keep the processor in standby longer, and makes it attractive to go into standby more often.

The Standby Design Exploration Space

Trade-off between different operational modes
Should blend smoothly with runtime optimizations

Slide 8.12
From the previous slides, a new version of our classic E-D trade-off curve emerges. The metrics to be traded off here are standby power versus wake-up delay.
Slide 8.13
Although standby modes are most often quoted for processors, they make just as much (if not more) sense for peripheral devices. Disks, wired and wireless interfaces, and input/output devices all operate in a bursty fashion. For instance, a mouse is in standby most of the time, and even when operational, data is only transmitted periodically. Clock gating and the support of different standby modes are hence essential. In this slide, the measured power levels and the transition times for two such peripheral devices are shown. Clearly the timing overhead associated with the wake-up from the standby mode cannot be ignored in each of these. Cutting down that time is crucial if standby is to be used more effectively.

Slide 8.14
Given the effectiveness of clock gating, there is little excuse for dynamic power dissipation in standby. Eliminating or drastically reducing standby currents is a lot more problematic. The main challenge is that contemporary CMOS processes do not feature a transistor that can be turned off completely.
Standby Static Power Reduction Approaches

- Transistor stacking
- Power gating
- Body biasing
- Supply voltage ramping

Increase the resistance in the leakage path, or reduce the voltage over that path. As the latter is harder to accomplish – you need either a variable or multiple supply voltages – most of the techniques presented in this chapter fall in the former category.

Transistor Stacking

- Off-current reduced in complex gates (see leakage power reduction @ design time)
- Some input patterns more effective than others in reducing leakage
- Effective standby power reduction strategy:
  - Select input pattern that minimizes leakage current of combinational logic module
  - Force inputs of module to correspond to that pattern during standby
- Pros: Little overhead, fast transition
- Con: Limited effectiveness

Slide 8.15
A standby leakage control technique must be such that it has minimal impact on the normal operation of the circuit, both from a functional and performance perspective. Lacking a perfect switch, only two leakage-reduction techniques are left to the designer:

Slide 8.16
In Chapter 4 we established that the stacking of transistors has a super-linear leakage reduction effect. Hence, it pays to ensure that the stacking effect is maximized in standby. For each gate, an optimal input pattern can be determined. To get the maximum effect, one has to control the inputs of each gate individually, which is unfortunately not an option. Only the primary inputs of a combinational block are controllable. Hence, the challenge is to find the primary input pattern that minimizes the leakage of the complete block. Even though stacking has a limited impact on the leakage, the advantage is that it virtually comes for free, and that it has a negligible impact on performance.
Slide 8.17
Standby leakage control using the stack effect requires only one real modification to the circuitry: all input latches or registers have to be presettable (either to the “0” or to the “1” state). This slide shows how this modification can be accomplished with only a minor impact on performance. Once the logic topology of a module is known, computer-aided design (CAD) tools can easily determine the optimal input pattern, and the corresponding latches can be inserted into the logic design.

Slide 8.18
Even when the technology-mapping phase of the logical-synthesis process is acutely aware of the stacking opportunity, it is unavoidable that some gates in the module end up with small fan-in. An inverter here or there is hard to avoid. And these simple gates contribute largely to the leakage. This can be remedied through the use of forced stacking, which replaces a transistor in a shallow stack by a pair (maintaining the same input loading). Although this transistor doubling, by necessity, impacts the performance of the gate – and hence should only be used in non-critical paths – the leakage reduction is substantial. This is perfectly illustrated by the leakage current (i.e., standby power) versus delay plots, shown on the slide for the cases of high- and low-threshold transistors. The advantage of forced stacking is that it can be fully automated.

Observe that this slide introduces another important trade-off metric: standby power versus active delay.
Power Gating

Disconnect module from supply rail(s) during standby
- Footer or header transistor, or both
- Most effective when high-\(V_{th}\) transistors are available
- Easily introduced in standard design flows
- But … Impact on performance

Very often called “MTCMOS” (when using high- and low-threshold devices)
(Ref: T. Sakata, VLSI’93; S. Mutoh, ASIC’93)

Slide 8.19
The ideal way to eliminate leakage current is to just disconnect the module from the supply rails — that is, if we could have perfect on-off switches available. The next best option is to use switches acting as “large resistors” between the “virtual” supply rails of the module and the global supply rails. Depending upon their position, those switches are called “headers” or “footers”, connecting to \(V_{DD}\) or ground, respectively. This power-gating technique performs the best when the technology supports both high- and low-threshold transistors. The latter can be used for the logic, ensuring the best possible performance, whereas the others are very effective as power-gating devices. When multiple thresholds are used, the power-gating approach is often called MTCMOS.

Power Gating – Concept

Leakage current reduces because
- Increased resistance in leakage path
- Stacking effect introduces source biasing

\[ V_{DD} \]
\[ \text{IN} = 0 \]
\[ \text{OUT} \]
\[ V_S = I_{leak}R_S \]
\[ \text{Sleep} \]

Slide 8.20
The headers/footers add resistance to the leakage path during standby. In addition, they also introduce a stacking effect, which increases the threshold of the transistors in the stack. The combination of resistance and threshold increase is what causes the large reduction in leakage current.
the input patterns. If one chooses a single power-gating device, the NMOS footer is the preferred option, because its on-resistance is smaller for the same transistor width. It can hence be sized smaller than its PMOS counterpart. This is the approach that is followed in a majority of today’s power-conscious IC designs.

**Slide 8.21**
Obviously, introducing an extra transistor in the charge and discharge paths of a gate comes with a performance penalty, the effects of which we would like to mitigate as much as possible. In principle, it is sufficient to insert only a single transistor (either footer or header) for leakage reduction. The addition of the second switch, though far less dramatic in leakage reduction, ensures that the stacking effect is exploited independent of the input patterns.

**Slide 8.22**
A number of modifications to the standard power-gating techniques can be envisioned, producing even larger leakage reductions, or reducing the performance penalty. The “boosted-gate” approach raises the gate voltage of the footer (header) transistors above the supply voltage, effectively decreasing their resistance. This technique is only applicable when the technology allows for high voltages to be applied to the gate. This may even require the use of thick-oxide transistors.

Some CMOS processes make these available to allow for the design of voltage-converting input and output pads (Note: the core of a chip often operates at a supply voltage lower than the board-level signals to reduce power dissipation).
**Slide 8.23**

The reverse is also possible. Instead of using a high-$V_{TH}$ device, the sleeper transistor can be implemented with a low-$V_{TH}$ device, leading to better performance. To reduce the leakage in standby, the gate of the sleeper is reverse biased. Similar to the “boosted-gate” technique, this requires a separate supply rail. Be aware that this increases the latch-up danger.

**Slide 8.24**

It is worth observing what happens with the virtual supplies in active and sleep modes. The extra resistance on the supply rail not only impacts performance, but also introduces extra $IR$-induced supply noise – impacting the signal integrity. During standby mode, the virtual supply rails start drifting, and ultimately converge to voltage levels determined by the resistive divider formed by the on and off transistors in the stack. The conversion process is not immediate though, and is determined by the leakage rates.

**Slide 8.25**

Reaching the standby operation mode is hence not immediate. This poses some interesting questions on where to put most of the decoupling capacitance (decap): on the chip supply rails, or on the virtual rails? The former has the advantage that relatively lower capacitance has to be (dis)charged when switching modes, leading to faster convergence and smaller overhead. The cost and overhead of going to standby mode is smaller. Also, the energy overhead for charging and discharging the decoupling capacitance is avoided. This approach
also has some important disadvantages: (1) the virtual supplies are more prone to noise, and (2) the gate-oxide capacitance that serves as decap stays under full voltage stress, and keeps contributing gate leakage current even in standby (Note that on-chip decoupling capacitance is often realized using huge transistors with their sources and drains short-circuited). Putting the decap on the chip supply rails hence is the preferred option if standby mode is most often invoked for a short time. The “decap on virtual supply” works best for long, infrequent standby invocations.

**Slide 8.26**
This trade-off is illustrated in the simulation of the virtual supply rails. After 10 ms, the leakage power of the “no decap on virtual rails” scenario has dropped by 90%. It takes 10 times as long for the “decap on virtual rails” to reach the same level of effectiveness.

**Slide 8.27**
As mentioned earlier, the sleep transistor does not come for free, as it impacts the performance of the module in active mode, introduces supply noise, and costs extra area. To minimize the area, a single switch is most often shared over a set of gates. An important question hence is how to size the transistor: making it wider minimizes performance impact and noise, but costs area. A typical target for sleep transistor sizing is to ensure that the extra ripple on the supply rail is smaller than 5% of the full swing.
How to Size the Sleep Transistor?

- Sleep transistor is not free – it will degrade the performance in active mode
- Circuits in active mode see the sleep transistor as extra power-line resistance
  - The wider the sleep transistor, the better
- Wide sleep transistors cost area
  - Minimize the size of the sleep transistor for given ripple (e.g., 5%)
  - Need to find the worst-case vector

Sleep Transistor Sizing

- High-$V_{TH}$ transistor must be very large for low resistance in linear region
- Low-$V_{TH}$ transistor needs less area for same resistance.

<table>
<thead>
<tr>
<th></th>
<th>MTCMOS</th>
<th>Boosted Sleep</th>
<th>Non-Boosted Sleep</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep TR size</td>
<td>5.1%</td>
<td>2.3%</td>
<td>3.2%</td>
</tr>
<tr>
<td>Leakage power reduction</td>
<td>1450x</td>
<td>3130x</td>
<td>11.5x</td>
</tr>
<tr>
<td>Virtual supply bounce</td>
<td>80 mV</td>
<td>59 mV</td>
<td>58 mV</td>
</tr>
</tbody>
</table>

[Ref: R. Krishnamurthy, ESSCIRC’02]

If the designer has access to power distribution analysis and optimization tools, sizing of the sleep transistors can be done automatically – as we will discuss in Chapter 12. If not, it is up to her to determine the peak current of the module through simulations (or estimations), and size the transistor such that the voltage drop over the switch is no larger than the allowable 5%.

Slide 8.28
The table on this slide compares the effectiveness of the different power-gating approaches. In the MTCMOS approach, a high-$V_{TH}$ device is used for the sleeper. To support the necessary current, the transistor must be quite large. When a low-$V_{TH}$ transistor is used, the area overhead is a lot smaller at the expense of increased leakage. The boosted-sleep mode combines the best of both, that is small transistor width and low leakage, at the expense of an extra supply rail. The transistors were sized such that the supply bounce for each of them is approximately the same.

Slide 8.29
The attentive reader must already have wondered about one important negative effect of power gating: when we disconnect the supply rails, all data stored in the latches, registers and memories of the module ultimately are lost. This sometimes is not a problem, especially when the processor always restarts from the same initial state – that is, all intermediate states can be forgotten. More often, the processor is expected to remember some part of its prior history, and rebooting from scratch after every sleep period is not an option. This can be dealt with in a number of ways:

- All essential-states are copied to memory with data retention before going to sleep, and is reloaded upon restart. Everything in the scratch-pad memory can be forgotten. The extra time for copying and reloading adds to the start-up delay.
Preserving State

- Virtual supply collapse in sleep mode causes the loss of state in registers
- Keeping the registers at nominal $V_{DD}$ preserves the state
  - These registers leak ...
- Can lower the $V_{DD}$ in sleep
  - Some impact on robustness, noise, and soft-error immunity

- The essential memory in the module is not powered down, but put in data retention mode. This approach increases the standby power, but minimizes the start-up and power-down timing overhead. We will talk more about memory retention in the next chapter.

- Only the logic is power-gated, and all registers are designed for data retention.

Latch-Retaining State During Sleep

transistors or gates that are shaded black. In essence, these are the transistors in the forward path, where performance is essential. The high-$V_{TH}$ cross-coupled inverter pair in the second latch acts as the data retention loop and is the only leakage-producing circuitry that is on during standby.

Slide 8.31

The above represents only a single example of a data retention latch. Many others have been perceived since then.

A very different option for state retention is to ensure that the standby voltage over the registers does not drop below the retention voltage (i.e., the minimum voltage at which the register or latch still reliably stores data). This is, for instance, accomplished by setting the standby voltage to $V_{DD}-V_D$ (where $V_D$ is the voltage over a reverse-biased diode), or by connecting it to a separate supply rail called $V_{retain}$. The former approach has the advantage that no extra supply is needed, whereas the latter allows for careful selection of the retention voltage so that leakage is minimized.
while retaining reliable storage. Both cases come with the penalty that the leakage through the logic may be higher than what would be obtained by simple power gating. The advantage is that the designer does not have to worry about state retention.

The topic of retention voltages and what determines their value is discussed in more detail in the next chapter on memory standby (Chapter 9).

Sleep Transistor Placement
To conclude the discussion on power gating, it is worth asking ourselves how this impacts layout strategy and how much area overhead this brings with it. Fortunately power switches can be introduced in contemporary standard layout tools with only minor variations.

In a traditional standard-cell design strategy, it is standard practice to introduce “strapper” cells at regular intervals, which connect the $V_{DD}$ and GND wires in the cells to the global power distribution network. These cells can easily be modified to include header and footer switches of the appropriate sizes. Actually, quite often one can determine the size of the switches based on the number of the cells they are feeding in a row.
Slide 8.33
The area overhead of the power-gating approach was quantified in a study performed at Intel in 2003 [Ref: J. Tschanz, ISSCC’03], which compared the effectiveness of various leakage control strategies for the same design (a high-speed ALU). Both footers and headers were used, and all sleep transistors were implemented using low-threshold transistors to minimize the impact on performance. It was found that the area overhead of the power gating was 6% for the PMOS devices, and 3% for the NMOS footers. We will come back to the same study in a couple of slides.

Slide 8.34
An alternative to the power-gating approach is to decrease the leakage current by increasing the thresholds of the transistors. Indeed, every transistor has a fourth terminal, which can be used to increase the threshold voltage through reverse biasing, which is a clear advantage. Recall that a linear change in threshold voltage translates into an exponential change in leakage current. Even better, this approach can also be used to decrease the transistor threshold in active mode through forward biasing! The alluring feature of dynamic biasing of the transistor is that it does not come with a performance penalty, and it does not change the circuit topology. The only drawback seems to be the need for a triple-well technology if we want to control both NMOS and PMOS transistors.

Although all this looks very attractive at a first glance, there are some other negatives that cannot be ignored. The range of the threshold control through dynamic biasing is limited, and, as we established in Chapter 2, it is rapidly decreasing with the scaling of the technology below 100 nm. Hence the effectiveness of the technique is quite small in nano-meter technologies, and will not get better in the future unless novel devices with much better threshold control emerge (for instance, the dual-gate transistors we briefly introduced in Chapter 2). Finally, changing the
back-gate bias of the transistors requires the charging or discharging of the well capacitance, which adds a sizable amount of overhead energy and time.

**Slide 8.35**
The concept of dynamic body biasing (DBB), as first introduced by Seta et al. in 1995, is illustrated pictorially in this slide. Obviously the approach needs some extra supply voltages that must be distributed over the chip. Fortunately, these extra supplies have to carry only little continuous current, and can be generated using simple on-chip voltage converters.

The technique of dynamic body biasing is by no means new, as it has been applied in memory designs for quite some time. Yet, it is only with leakage power becoming an important topic that it is being applied to computational modules. The attentive reader probably realizes that this technique has more to offer than just leakage management. It can, for instance, also be used for the compensation of threshold variations. To hear more about this, you have to wait for the later chapters.

**Slide 8.36**
Though the adoption of DBB requires little changes in the computational modules, it takes some extra circuitry to facilitate the switching between the various biasing levels, which may extend above or below the standard voltage rails. Adapting the sleep control signals (CE) to the appropriate levels requires a set of level converters, whose outputs in turn are used to switch the well voltages. The resulting voltage waveforms, as recorded in [Seta95], are shown on the slide as well. Observe that in this early incarnation of the DBB approach it takes approximately the same time to charge and discharge the wells – for a total transient time of somewhat less than 100 ns.
Slide 8.37
The area overhead of the dynamic-biasing approach mainly consists of the generation of the bias voltages, the voltage switches, and the distribution network for the bias voltages. To compare DBB with power gating, the example of Slide 8.33 is revisited. The body-bias circuitry consists of two main blocks: a central bias generator (CBG) and many distributed local bias generators (LBGs). The function of the CBG is to generate a process-, voltage-, and temperature-invariant reference voltage, which is then routed to the LBGs. The CBG uses a scaled-bandgap circuit to generate a reference voltage, which is 450 mV below the main supply — this represents the amount of forward bias to apply in active mode. This reference voltage is then routed to all the distributed LBGs. The function of the LBG is to refer the offset voltage to the supply voltages of the local block. This ensures that any variations in the local supplies will be tracked by the body voltage, maintaining a constant 450 mV of FBB.

To ensure that the impedance presented to the well is low enough, the forward biasing of the ALU required 30 LBGs. Observe that in this study only the PMOS transistors are dynamically biased, and that only forward biasing is used (in standby, zero bias is used). The total area overhead of all the bias units and the wiring turned out to be approximately 8%.

Slide 8.38
The effectiveness of the DBB approach is demonstrated with an example of an application-specific processor, the SH-mobile from Renesas (also called the SuperH Mobile Application Processor). The internal core of the processor operates at 1.8 V (for a 250 nm CMOS technology). In standby mode, reverse body-biasing is applied to the PMOS (3.3 V) and the NMOS (−1.5 V) transistors. The 3.3 V supply is already externally available for the

---

**Application-specific processor (SH-mobile)**
- 250 nm technology
- core at 1.8 V
- I/O at 3.3 V
- 3.3 V transistors

[Ref. M. Miyazaki, Springer 06]
I/O pins, whereas the $-1.5$ V supply is generated on-chip. Similar to the power-gating approach, special “switch cells” are included in every row of standard cells, providing the circuitry to modulate the well voltages.

For this particular design, the DBB approach reduces the leakage by a factor of 28 for a fairly small overhead. Unfortunately, what works for 250 nm does not necessarily translate into similar savings in the future.

Effectiveness of Dynamic Body Biasing

![Graph showing effectiveness of Dynamic Body Biasing](image)

Practical $V_{TH}$ tuning range less than 150 mV in 90 nm technology

Slide 8.39
As we had already observed in Slide 2.12, the effectiveness of back biasing reduces with technology scaling. Although for a 90 nm technology, a combination of FBB and RBB may still yield a 150 mV threshold change, the effect is substantially smaller for 65 nm. This trend is not expected to change course substantially in the coming technology generations. The potential savior is the adoption of dual-gate devices, which may be adopted in the 45 nm (and beyond) technology generations. Hence, as for now, DBB is a useful technology up to 90 nm, but its future truly depends upon device and manufacturing innovations.

Supply Voltage Ramping (SVR)

- Reduce supply voltage of modules in sleep mode
  - Can go to 0 V if no state-retention is necessary
  - Down to state retention voltage otherwise, (see Memory in next chapter), or move state to persistent memory before power-down
- Most effective leakage reduction technique
  - Reduces current and voltage
- But
  - Needs controllable voltage regulator
    - Becoming present more often in modern integrated system designs
  - Longer reactivation time

Simplified version switches between $V_{DD}$ and GND (or $V_{DDL}$)

[Ref: M. Sheets, VLIS'06]

Slide 8.40
Ultimately, the best way to reduce leakage in standby mode is to ramp the supply voltage all the way down to 0 V. This is the only way to guarantee the total elimination of leakage. A controllable voltage regulator is the preferred way of accomplishing this Supply Voltage Ramping (SVR) scheme. With voltage islands and dynamic voltage scaling becoming common practice (see Chapter 10), voltage regulators and converters are being integrated into the SoC design process, and the overhead of SVR is negligible. In designs where this is not the case, switches can be used to swap the “virtual” supply rail between $V_{DD}$ and GND. As the switches themselves leak, this approach is not as efficient as the ramping.
The overhead of the SVR scheme is that upon reactivation all the supply capacitance has to be charged up anew, leading to a longer start-up time. Obviously all state data are lost in this regime. If state retention is a concern, techniques discussed earlier in the chapter such as the transfer of essential state information to persistent memory or keeping the supply voltage of the state memory above the retention voltage (DRV), are equally valid.

**Slide 8.41**
This slide shows a pictorial perspective of the supply ramping approach (both for ramping down to GND or to the data retention voltage DRV). SVR in concert with dynamic voltage scaling (DVS – see Chapter 10) is at the core of the “voltage island” concept, in which a chip is divided into a number of voltage domains that can change their values dynamically and independently. To have the maximum effect, it is important that signals crossing the boundaries of voltage islands are passed through adequate converters and isolators. This is in a sense similar to the boundary conditions that exist for signals crossing clock domains.

**Slide 8.42**
The impact of SVR is quite important. Because of the exponential nature of the DIBL effect, just reducing the supply voltage from 1 V to 0.5 V already reduces the static leakage power by a factor of 8.5 for a four-input NAND gate in a 90 nm CMOS technology. With the proper precautions (as we will discuss in Chapter 9), data retention may be ensured all the way to 300 mV. However, nothing beats scaling down all the way to ground.
Integration in Standard-Cell Layout Methodology

- Power switch cell easily incorporated into standard design flow
  - Cell has same pitch as existing components
  - No changes required to cell library from foundry
- Switch design can be independent of block size

Slide 8.43
If voltage ramping is not an option, switching between different rails (from $V_{DDH}$ to GND or $V_{DDL}$) is still a viable alternative, even though it comes with a larger leakage current in standby (through the $V_{DDH}$ switch). The switch to the lower rail ($V_{DDL}$ or GND) can be made small as it only carries a very small amount of current. The SVR approach can be incorporated in the standard design flows in a similar way as the power-gating and the dynamic body-biasing approaches discussed earlier. The only requirement is a number of extra cells in the library with appropriate sizing for the different current loads.

Slide 8.44
Though standby power reduction is a major challenge, it is also clear that a number of techniques have emerged addressing the problem quite effectively as shown in this overview slide. The main trade-offs are between standby power, invocation overhead, area cost, and runtime performance impact.

The main challenge the designer faces is to ensure that a module is placed in the appropriate standby mode when needed, taking into account the potential savings and the overhead involved. This requires a system-level perspective, and a chip architecture with power management integrated into its core.

Slide 8.45
In the end, what is really needed to deal with the standby power problem is an ideal switch, which conducts very small current when off and has a very low resistance when on. Given the importance of standby power, spending either area or manufacturing cost toward such a device seems to be worthwhile investment. In Chapter 2, we discussed some emerging devices that promise steeper sub-threshold slopes, such as, for instance, the dual-gate device. Some speculative transistors even promise slopes lower than 60 mV/dec.
Yet, a switch that can be fully turned off in standby mode would be the ultimate. This is why some of the current research into micro-electromechanical systems (MEMS)-based switches is so appealing. A number of research groups are investigating the idea of a MOS transistor with a "movable" gate, where the thickness of the gate insulator is modified using electrostatic forces. This may lead to a switch with ignorable leakage in off mode, yet a good conductivity in the on mode. We believe that devices like this may ultimately play an important role in extending the life of CMOS design into the nanometer scale. Yet, as always, any modification in the manufacturing process comes at a considerable cost.

**Summary and Perspectives**

- Today’s designs are not leaky enough to be truly power-performance optimal! Yet, when not switching, circuits should not leak!
- Clock gating effectively eliminates dynamic power in standby
- Effective standby power management techniques are essential in sub-100 nm design
  - Power gating the most popular and effective technique
  - Can be supplemented with body biasing and transistor stacking
  - Voltage ramping probably the most effective technique in the long range (if gate leakage becomes a bigger factor)
- Emergence of “voltage or power” domains

In the next chapter, we shall discuss how the inverse is true for memories, where controlling standby leakage while retaining storage has evolved into one of the most challenging problems for today and tomorrow.
References

Books and Book Chapters


Articles

- T. Kuroda et al., "A 0.9V 150 MHz 10 mW 4 mm² 2-D discrete cascode transistor core processor with variable-threshold-voltage scheme," JSSC, 31(11), pp. 1770–1779, Nov. 1996.

References (cont.)

Chapter 9
Optimizing Power @ Standby – Memory

Slide 9.1
This chapter describes approaches for optimizing the power consumption of an embedded memory, when in standby mode. As mentioned in Chapter 7, the power dissipation of memories is, in general, only a fraction of the overall power budget of a design in active mode. The reverse is true when the circuit is in standby. Owing to the large (and growing) number of memory cells on a typical IC, their contribution to the leakage power is substantial, if not dominant. Reducing the standby power dissipation of memories is hence essential.

Slide 9.2
In this Chapter, we first discuss why the standby leakage current of large embedded SRAM memories is becoming a growing concern. When looking at the possible solution space, it becomes clear that static power in the memory core is best contained by manipulating the various voltages in and around the cell. One option is to reduce the supply voltage(s); another is to change the transistor bias voltages. Various combinations of these two can be considered as well. Bear in mind however that any realistic leakage power reduction technique must ensure that the data is reliably retained during the standby...
period. Though the periphery presents somewhat of a lesser challenge, it has some special characteristics that are worth-examining. The Chapter is concluded with some global observations.

**Slide 9.3**

During standby mode, an embedded memory is not accessed, so its inputs and outputs are not changing. The main function of the memory during standby is therefore to retain its data until the next transition to active operation. The retention requirement complicates the reduction of the leakage power. Although combinational logic modules can be disconnected from the supply rails using power gating, or their supply voltages reduced to zero, this is not an option for the embedded SRAM (unless it is a scratch-pad memory). Hence, minimizing the leakage, while reliably maintaining state, is the predominant requirement. Some of the techniques that are introduced in this chapter carry some overhead in terms of power and/or time to bring a memory in to and/or out of standby. A secondary metric is hence the energy overhead consumed during the transition, which is important because it determines the minimum time that should be spent in standby mode for the transition to be worthwhile. If the power savings from being in standby for some time do not offset the overhead of entering/leaving that mode, then standby should not be used. In addition, rapid transitions between standby and active modes are helpful in many applications. Finally, we also observe that reducing standby power often comes with an area overhead.

We begin this chapter by taking a brief top-level look at the operation of an embedded SRAM cell during standby. Next, we examine a number of standby power reduction techniques. The most effective techniques to date are based on voltage manipulation—either lowering the supply voltage, or increasing the bias voltages of the transistors inside the cell. The standby power of the peripheral circuits is briefly discussed before the chapter is summarized.
Reminder of “Design-Time” Leakage Reduction

- Design-time techniques (Chapter 7) also impact leakage
  - High-$V_{TH}$ transistors
  - Different precharge voltages
  - Floating BLs
- This chapter: adaptive methods that uniquely address memory standby power

leakage power during standby, this chapter focuses on approaches that uniquely address the standby leakage.

The Voltage Knobs

- Changing internal voltages has different impact on leakage of various transistors in cell
- Voltage changes accomplished by playing tricks with peripheral circuits

[Ref: Y. Nakagome, IBM’03]

Slide 9.4
Some of the approaches described in Chapter 7 for lowering power at design time reduce leakage power in both active and standby modes. These approaches include using high-threshold-voltage transistors, lowering the precharge voltage, or allowing bitlines to float (they float to a voltage that minimizes leakage into the bitcells. Though these approaches do affect the

Slide 9.5
Though there are many circuit-level knobs available for addressing leakage power, the various voltage levels in and around the bitcell are the most effective. In Chapter 7, we discussed how these voltages can be assigned at design time to reduce power. Altering the voltages by manipulating the peripheral circuits during standby mode can decrease leakage power during standby mode. There is more flexibility to alter the voltages in standby mode because many of the functionality-limiting metrics are no longer relevant, such as read static noise margin and write margin. In standby mode, the primary functionality metric of concern is the hold static noise margin, as the bit-cells are only holding their data.

Slide 9.6
The most straightforward voltage scaling approach to lowering standby leakage power in a memory is reducing the supply voltage, $V_{DD}$. This approach lowers power in two ways: (1) voltage reduction ($P = VI$) and (2) leakage current reduction. The dominant mechanism behind the latter is the drain-induced barrier lowering (DIBL) effect. In addition, other contributors to leakage current drop off as well. Gate-induced drain leakage (GIDL) quickly decreases with $V_{DD}$, and
Lower $V_{DD}$ in Standby

- Basic Idea: Lower $V_{DD}$ lowers leakage
  - sub-threshold leakage
  - GIDL
  - gate tunneling
- Question: What sets the lower limit?
  [Ref: K. Flautner, ISCA’02]

active mode, the power supply returns to the nominal operating voltage. As we have described before, the key limitation to the extent by which $V_{DD}$ is lowered is that the data inside the cells must be protected. If the data are no longer required, then the power supply can simply be disconnected using power gating approaches like those that were described earlier for combinational logic, or by ramping the supply down to GND.

Slide 9.7
Given the effectiveness of voltage reduction in lowering the standby power of an SRAM memory, the ultimate question now is how much the supply voltage can safely be reduced. We define the minimum supply voltage for which an SRAM bit-cell (or an SRAM array) retains its data as the Data Retention Voltage (DRV).

The butterfly plots shown on this slide illustrate how the noise margins of a 6T cell (with its access transistors turned off) collapse as the supply voltage is reduced. Due to the asymmetrical nature of a typical cell (caused by the dimensioning of the cell transistors as well as by variations), the SNM of the cell is determined by the upper lobe of the butterfly plot. Once the supply voltage reaches 180 mV, the SNM drops to zero and the stored value is lost. The cell becomes monostable at that point. In a purely symmetrical cell, the supply voltage could be lowered substantially more before the data is lost.
We can therefore also specify the DRV as the voltage at which the SNM of a non-addressed cell (or cell array) drops to zero.

**Slide 9.8**
The advantages of scaling the $V_{DD}$ during standby can be quite significant. A 0.13 $\mu$m test chip shows over 90% reduction in standby leakage by lowering the power supply to within 100 mV of the DRV. The reduction in the DIBL effect is one of the most important reasons behind this large drop in leakage current.

Hence, it seems that minimizing the DRV voltage of a memory is an effective means to further reductions in standby leakage power.

**Slide 9.9**
The DRV of a bit-cell depends upon a range of parameters. Intuitively we can see that the DRV would be minimized if its butterfly curve would be symmetrical — that is, that the upper and lower lobes should be of equal size. This is accomplished if the pull-up and pull-down networks (including the turned-off NMOS access transistors) are of equal strength.

From this, it becomes clear that the DRV must be a function of the sizes of transistors in the bit-cell. As the DRV voltage typically lies below the threshold voltage of the process, it means that all transistors operate in the sub-threshold mode. Under these operational conditions, the standard (strong-inversion) rationing rules between NMOS and PMOS transistors do not apply. In strong inversion, NMOS transistors are typically 2–3 times stronger than equal-sized PMOS devices owing to the higher electron mobility. In the sub-threshold region, the relative strength is determined by the leakage current parameter $I_s$, the threshold voltage $V_{TH}$, and the sub-threshold slope factor $n$ of the respective devices. In fact, sub-threshold PMOS transistors may be substantially stronger than their NMOS counterparts.
The influence of changing the respective transistor sizes on a generic 6T cell is shown in the slide. For this cell, increasing the size of the PMOS transistors has the largest impact on DRV. Given the strong pull-down/weak pull-up approach in most of the generic cells, this is not unexpected.

Note: Though a symmetrical butterfly curve minimizes the DRV voltage, it is most likely not the best choice from an active read/write perspective. SRAM memories provide fast read access through precharged bitlines and strong NMOS discharge transistors. This leads automatically to an asymmetrical cell.

Slide 9.10
Any variation from the symmetrical bit-cell causes a deterioration of the DRV. This is illustrated in this slide where the impact of changing the relative strengths of the sub-threshold transistors is shown. Both strong NMOS (S\(_N\)) and strong PMOS (S\(_P\)) transistors warp the butterfly curves and reduce the SNM.

Slide 9.11
Given the high sensitivity of the DRV to the relative strengths of transistors, it should be no surprise that process variations have a major impact on the minimal operational voltage of an SRAM cell. Local variations in channel length and threshold voltages are the most important cause of DRV degradation. This is best demonstrated with some experimental results. This plot shows a 3-D rendition of the DRV of a 130 nm 32 Kb SRAM memory, with the x- and y-axis indicating the position of the cell in the array, and the z-axis denoting the value of the DRV. Local transistor variations seem to cause the largest DRV changes. Especially threshold variations play a major role.
The histogram of the DRV values shows a long tail, which means that only a few cells exhibit very high values of the DRV. This is bad news: the minimum operation voltage of a complete memory (that is, the DRV of the complete memory) is determined by the DRV of the worst-case cell, padded with some extra safety margin. This means that the DRV of this particular memory should be approximately 450 mV (with a 100 mV safety margin added), though most of the cells operate perfectly well even at 200 mV.

**Slide 9.12**
A similar picture emerges for memories implemented in the 90 nm and 45 nm (in this particular case, a 5 Kb memory). Clearly, local variations cause a DRV distribution with a long tail toward higher DRVs, and the influence of local variations increases with process technology scaling.

The DRV also depends on other factors (but less strongly so) such as global variations, the stored data values, temperature, and the bitline voltage.

**Slide 9.13**
Understanding the statistical distribution of the DRV is a first and essential step toward identifying which circuit techniques would be most effective in lowering operational voltage and hence leakage. (This will be painstakingly made clear in Chapter 10, where we discuss runtime power reduction techniques).

Inspection of the DRV distribution shows that it follows neither a normal nor a log-normal model. A better match is presented by the equation shown in the slide. The resulting model matches true Monte-Carlo simulation along the DRV tail to 6σ – which means that outliers can be predicted quite effectively. The
independent parameters of the model ($\mu_0$ and $\sigma_0$ – the mean and variance of the SNM at a supply voltage $V_0$) can be obtained from a small Monte-Carlo simulation (at $V_{DD} = V_0$) of the SNM in the one lobe of the butterfly plot that is the most critical.

Slide 9.14

Building on the presented analysis of the DRV, its parameters and its statistics, we can devise a number of strategies to lower its value. The first approach is to use optimization. The available options are appropriate sizing of the transistors to either balance the cell or reduce the impact of variations, careful selection of the body-biasing voltages (for the same reasons), and/or playing with the peripheral voltages to compensate for unbalancing leakage currents. The net effect is to shift the DRV histogram to the left. Most importantly, the worst-case value is also lowered as is indicated in green on the chart. Nothing ever comes for free though – manipulating the DRV distribution means trading off some other metric such as area or access time. The designer must therefore weigh the importance of DRV for low-power standby mode with other design considerations.

A second approach is to lower the voltage below the worst-case value. This approach, which we will call “better-than-worst-case” design in the next chapter, may potentially lead to errors. As the tail of the distribution is long, the number of failing cells will be relatively small. The addition of some redundancy in the form of error detection can help to capture and correct these rare errors. Error-correcting (ECC) strategies have been exploited for a long time in DRAM as well as in non-volatile memories, but are not a common practice in embedded SRAMs. The potential benefits in leakage reduction and overall robustness are worth the extra overhead. From a DRV perspective, the impact of ECC is to lob off the tail of DRV distribution (as indicated in red).

Naturally, both cell optimization and ECC can be applied in concert resulting in a DRV with a lower mean and narrower distribution (indicated in blue).
Slide 9.15
ECCs have been used in memories for a long time. Already in the 1970s, ECC had been proposed as a means to improve the yield of DRAMs. Similarly, error correction is extensively used in Flash memories to extend the number of write cycles. As indicated in the previous slides, another use of ECC is to enable “better-than-worst case”, and lower the supply voltage during standby more aggressively.

The basic concept behind error detection and correction is to add some redundancy to the information stored. For instance, in a Hamming (31, 26) code, five extra parity bits are added to the original 26 data bits, which allows for the correction of one erroneous bit (or the detection of two simultaneous errors). The incurred overhead in terms of extra storage is approximately 20%. Encoder and decoder units are needed as well, further adding to the area overhead. The leakage current reduction resulting from the ECC should be carefully weighed against the active and static power of the extra cells and components.

Yet, when all is considered, ECC yields substantial savings in standby power. Up to 33% in leakage power reduction can be obtained with Hamming codes. Reed–Muller codes perform even a bit better, but this comes at the cost of a more complex encoder/decoder and increased latency.

Slide 9.16
The impact of combining cell optimization and error correction is illustrated for a 26 Kb SRAM memory (implemented in a 90 nm CMOS technology). The use of a (31, 26, 3) Hamming code actually increases the total size of the memory to 31 Kb.

The optimized memory is compared with a generic implementation of the memory, integrated on the same die. In all scenarios, a guard band of 100 mV above the minimum allowed DRV is
maintained. The DRV histograms illustrate how the combination of optimization and ECC both shifts the mean DRV to lower values and narrows the distribution substantially. The leakage current in standby is reduced by a factor of 50. This can be broken down as follows:

- Just lowering the supply voltage of the generic memory to its DRV + 100 mV reduces the leakage power by 75%.
- Optimizing the cell to lower the DRV yields another 90% reduction.
- Finally, the addition of ECC translates into an extra 35% savings.

For this small memory, the area penalty to accomplish this large leakage savings is quite substantial. The combination of larger cell area, additional parity bits, and encoders and decoders approximately doubles the size of the memory. Though this penalty may not be acceptable for top-of-the-line microprocessor chips with huge amounts of cache memory, the reduction in static power makes this very reasonable in ultra low-power devices with low duty cycles (such as those encountered in wireless sensor networks or implanted medical devices).

**Slide 9.17**
The standby voltage reduction techniques discussed so far lower the supply voltage to a value that is set at a guard band above the worst-case DRV. The latter is obtained by careful modeling, simulation, and experimental observation of the process variability. This open-loop approach means that all chips that do not suffer the worst-case DRV cannot take full advantage of the potential leakage savings. It has been widely reported that the difference in leakage current between the best- and worst-case instances of the same design can vary by as much as a factor of 30.

One way to get around this is to use a closed-loop feedback approach, which promises to increase the leakage savings for every chip. The idea is to measure the distributions on-line, and set the standby voltage accordingly. The measurements are provided by a set of “canary replica cells” added to the memory (as in “the canary in the coal mine” strategy used to detect the presence of noxious fumes in mines in older times).

The canary cells are intentionally designed to fail across a range of voltages above the DRV distribution of the core SRAM cells. Based on the knowledge of the shape underlying the SRAM-cell DRV distribution (using models such as the one presented in Slide 9.13), the feedback loop uses the measured data to dynamically set the supply voltage.

The diagram shows a prototype architecture that organizes the canary cells in banks. Each canary cell is structured and sized like the bit-cells in the core array, except that an additional PMOS header switch presents a lower effective $V_{DD}$ to the cell. Controlling the gate voltage of the PMOS headers ($V_{CTRL}$) allows us to set the DRV of the canary cells across a wide range of voltages.

**Note:** The “canary” approach is a first example of a runtime power reduction technique, which is the topic of Chapter 10.
How to Approach the DRV Safely?

Canary Replica &
Multiple sets of
test circuit

failure
Threshold

SRAM cell

0.6% area overhead in 90 nm test chip

[Ref: J. Wang, CICC’07]

Slide 9.18
The concept of how canary cells can be used to estimate the “safe” operational voltage is illustrated in the top left drawing. The cells are divided into clusters, tuned to fail at regular intervals above the average DRV of the core cells. To reduce the spread of DRV distribution of the canary cells relative to the core, larger sizes are used for canary transistors. Of course, the small set of canary cells cannot track the local variations in the main array, but it is sufficient to estimate the global ones (such as systematic variations or the impact of temperature changes), and hence remove a large fraction of the guard band.

By varying $V_{CTRL}$ (e.g., by providing different values of $V_{CTRL}$ to different banks) and measuring the failure point, an estimate of the safe value of the minimum operational voltage is obtained. The measured relationship between the DRV of the canary cells and $V_{CTRL}$ is shown in the lower-left plot, demonstrating clearly that $V_{CTRL}$ is a good measure for the DRV value.

A 90 nm test chip implements the canary-based feedback mechanism at a cost of 0.6% area overhead. Measurements confirm that the canary cells reliably fail at voltages higher than the average core cell voltage and that this relationship holds across environmental changes. This approach helps to reduce leakage power by factors of up to 30 compared to a guard band approach.

Slide 9.19
All standby power reduction techniques discussed so far are based on lowering the $V_{DD}$. An alternative approach is to raise the ground node of the bit-cells, $V_{SS}$. This approach decreases $V_{DS}$ across a number of transistors, which lowers sub-threshold conduction (due to DIBL) as well as the GIDL effect. Furthermore, for bulk NMOS devices, the higher $V_{SS}$ causes a negative $V_{BS}$ that increases

Raising $V_{SS}$

- Raise bit-cell $V_{SS}$ in standby (e.g., 0 to 0.5 V)
- Lower BL voltage in standby (e.g., 1.5 to 1 V)

‘0’ is 0.5 V

Lower voltage $\rightarrow$ less gate leakage and GIDL

Lower $V_{DS}$ $\rightarrow$ less sub-$V_{TH}$ leakage (DIBL)

Negative $V_{BS}$ reduces sub-$V_{TH}$ leakage

[Ref: K. Osada, JSSC’03]
the threshold voltage of the transistors, and lowers the sub-threshold current exponentially. The cell presented in this slide exploits all of these effects. The choice between raising $V_{SS}$ and lowering $V_{DD}$ depends primarily on the dominant sources of leakage in a given technology and on the relative overhead of the different schemes.

**Slide 9.20**

Another option is to intentionally apply reverse body biasing (RBB) to the transistors in the cell during standby mode. Again, an increase in threshold voltage translates into an exponential decrease in sub-threshold drain–source leakage current, which makes it a powerful tool for lowering standby currents.

To induce RBB, you can either raise the source voltage (as in raised-$V_{SS}$ approach of Slide 9.19) or lower the body voltage for an NMOS. In traditional bulk CMOS, modulating the NMOS body node means driving the full capacitance of the P-type substrate. Transitioning in and out of standby mode hence comes with a substantial power overhead. Changing the body voltage of the PMOS is relatively easier because of the smaller-granularity control offered by the N-well. Many bulk technologies now offer a triple-well option that allows for the placement of NMOS transistors in a P-well nested inside an N-well. This option makes adjustable RBB for standby mode more attractive, but the energy involved in changing the voltage of the wells must still be considered.

This slide shows an RBB scheme that raises and lowers the PMOS and NMOS bulk voltages, respectively, whenever a row is not accessed. The advantage of this approach is that it operates at a low level of granularity (row-level), in contrast to all techniques discussed previously, which work on a per-block level. In general, at most a single row of a memory module is accessed at any given time. The penalty is an increase in read and write access times.
Combining Body Biasing and Voltage Scaling

Body biasing is a technique that can easily be deployed in conjunction with other standby power reduction methods. This slide, for example, shows an SRAM that combines body biasing and supply voltage scaling. During active mode, the $V_{DD}$ and $V_{SS}$ rails for the accessed cells are set at slightly more positive and negative, respectively, than during standby. At the same time, the body terminals of the transistors are driven to 0 and $V_{DD}$ such that the cell transistors have a slight forward body bias (FBB). The reduced $V_{TH}$ improves the read/write access times. In standby mode, the power rails are pinched inward and RBB is applied. The combination of voltage scaling and body bias potentially provides for a dramatic reduction in standby power. However, one has to ensure that the double overhead of supply and body voltage scaling does not offset the gains. Also, one has to make sure that the source/drain diodes are not forward biased in FBB mode.

Combining Raised $V_{SS}$ and RBB

Slide 9.22
Similarly we can combine the raised-$V_{SS}$ approach with RBB. During standby, the raised-$V_{SS}$ node reduces the effective $V_{SS}$ node voltage of the cell, while providing RBB for the NMOS transistors. A raised N-well voltage provides RBB to the PMOS devices. The advantage of this approach is that a triple-well technology is not required.

Slide 9.23
From Chapters 7 and 9 emerges a wide spectrum of choices in setting the voltages in SRAMs during active and standby modes. The design parameters include the choice of not only the supply and well voltages, but also the peripheral voltages such as wordline and bitline voltages. A literature survey illustrates the broad range of options available.
In essence, each of these approaches follows the same principles:

- For each operational mode, voltage values are selected to minimize power while ensuring functionality and reliability. The latter means that noise margin and DRV constraints must be met in active and standby mode, respectively. In addition, the impact on read and write access times as well as on memory area must be kept within bounds.

- Transition between modes often means that multiple voltages must be adopted. The overhead in time and power of these transitions should be carefully weighed against the gains.

Anyone who has ever designed SRAMs knows that the impact of a change in the cell or the periphery can be quite subtle. Although the different techniques presented here may seem to yield huge benefits, a careful analysis including intensive simulation and actual prototyping is absolutely essential in defining the ultimate benefits and pitfalls.

**Slide 9.24**

As we had mentioned in Chapter 7, the peripheral circuits that go around the SRAM array primarily consist of combinational logic (examples are the write drivers, row and column decoders, I/O drivers). Most of these circuits can be disabled during standby mode, and their leakage can be reduced using the techniques from Chapter 8. However, there are some characteristics of the SRAM periphery circuits that differentiate them from generic logic and thus deserve mentioning.

- Although the majority of transistors in an SRAM are situated in the memory array, the SRAM periphery can still contribute a sizable amount of leakage. This can be attributed to the fact that most components of the periphery must be sized fairly large to drive the large capacitances inside the array (e.g., wordline and bitlines). These wide transistors come with large leakage currents.
• Whereas the SRAM bit-cells typically use high-threshold transistors, performance considerations dictate the use of low-threshold devices in the periphery.
• From our discussion, it had become clear that memory cells and logic are on somewhat different voltage-scaling trajectories. Logic supply voltages are expected to keep on scaling downward, whereas reliability concerns in the presence of increasing process variations force the voltages in memory to stay constant (if not increasing). Interfacing between periphery and memory core hence increasingly requires the presence of voltage-up and -down converters, which translates into a timing and power overhead. Moreover, this interface must be properly conditioned in standby mode. For example, floating wordlines caused by power gating of the periphery, could cause data loss in the bit-cells.

On the other hand, a sizable number of the generic standby power management techniques introduced in Chapter 8 perform even better when applied to memory periphery. This is largely due to the well-behaved repetitive structure of the peripheral circuits. In addition, many of the signal voltages during standby are known very well. For instance, we know that all of the wordlines must be 0 in standby. This knowledge makes it easier to apply power gating or forced stacking to maximally reduce the leakage power.

<table>
<thead>
<tr>
<th>Summary and Perspectives</th>
</tr>
</thead>
<tbody>
<tr>
<td>▪ SRAM standby power is leakage-dominated</td>
</tr>
<tr>
<td>▪ Voltage knobs are effective to lower power</td>
</tr>
<tr>
<td>▪ Adaptive schemes must account for variation to allow outlying cells to function</td>
</tr>
<tr>
<td>▪ Combined schemes are most promising</td>
</tr>
<tr>
<td>- e.g., Voltage scaling and ECC</td>
</tr>
<tr>
<td>▪ Important to assess overhead!</td>
</tr>
<tr>
<td>- Need for exploration and optimization framework, in the style we have defined for logic</td>
</tr>
</tbody>
</table>

Slide 9.25
In summary, SRAM leakage power is a dominant component of the overall standby power consumption in many SoCs and general-purpose processing devices. For components that operate at low duty cycles, it is often THE most important source of power consumption. In this chapter, we have established that the most effective knobs in lowering leakage power are the various voltages that drive the bit-cells. However, these voltages must be manipulated carefully so that data preservation is not endangered.

As with active operation, the large number of small transistors in an embedded SRAM means that the far tails of power and functionality distributions drive the design. This means that any worst-case or adaptive schemes must account for the outliers on the distributions to preserve proper SRAM functionality. The most promising schemes for leakage reduction combine several different voltage-scaling approaches (selected from the set of $V_{TH}$, $V_{DD}$, $V_{SS}$, and well and periphery voltages) along with architectural changes (e.g., ECC). In all of these approaches, the overhead requires careful attention to ensure that the overall leakage savings are worth the extra cost in area, performance, or overhead power.

All this having been said, one cannot escape the notion that some more dramatic steps may be needed to improve the long-term perspectives of on-chip memory. Non-volatile memory structures that are compatible with logic processes and that do not require high voltages present a promising venue. Their non-volatile nature effectively eliminates the standby power concern. However, their write and (sometimes) read access times are substantially longer than what can be obtained with SRAMs. It is worth keeping an eye on the multitude of cell structures that are currently trying to make their way out of the research labs.
References

Books and Book Chapters:

Articles:

References (cont.)

Chapter 10
Optimizing Power @ Runtime – Circuits and Systems

Slide 10.1
The computational load and hence the activity of a processor or an SoC may change substantially over time. This has some profound repercussions on the design strategy for low power, as this means that the optimal design point changes dynamically. The standby case, discussed in the previous chapters, is just a special case of these dynamic variations (with the activity dropping to zero). The concept of runtime optimization in the energy–delay space presents a fundamental departure from traditional design methodology, in which all design parameters such as transistor sizes and supply and threshold voltages were set by the designer or the technology, and remained fixed for the lifetime of the product. Though runtime optimization creates some unique opportunities, it also presents some novel challenges.
Slide 10.2
The Chapter starts by motivating the need for dynamic adaptation. A number of different strategies to exploit the variation in activity or operation mode of a design are then described in detail. Dynamic voltage- and body-bias scaling are the best-known examples. Increasingly, it becomes necessary to dynamically adjust a broad range of design parameters, leading to a self-adapting approach. In the extreme case, one can even adjust the design outside the safe operation zone to further save energy. This approach is called “aggressive deployment” or “better than worst-case” design. Finally, we discuss how the adoption of these runtime techniques leads to the need for a power management system, or, in other words, “a chip operating system”.

Slide 10.3
Activity variations and their impact on power are an important reason why runtime optimization had become an attractive idea in the late 1990s. Since then, other important sources of dynamic variations have emerged. Device parameters change over time owing to aging or stress effects, or owing to varying environmental conditions (e.g., temperature). Changes in current loads cause the supply rails to bounce up and down. These added effects have made runtime optimization over the available design parameters even more attractive. Sticking to a single operational point is just too ineffective.

Slide 10.4
To illustrate just how much workloads can vary over time, let us consider the case of a video compression module. A fundamental building block of virtually every compression algorithm is the motion compensation block, which computes how much a given video frame differs from the previous one and how it has changed. Motion compensation is one of the most computationally intensive functions in video compression algorithms such as MPEG-4 and H.264.
One can intuitively understand that the motion compensation module has to work a lot harder in a fast-moving car chase scene than in a slow pan of a nature landscape. This is clearly illustrated in the chart of the lower-right corner, which plots a histogram of the number of IDCTs (inverse discrete cosine transforms) that have to be performed per frame. The distribution is strongly bi-modal. It also shows that the computational effort per frame can vary over 2–3 orders of magnitude.

Slide 10.5
The same broad distribution holds for general-purpose computing as well. Just watch the “CPU Usage” chart of your laptop for a while. Most of the time, the processor runs at about 2–4% utilization, with occasional computational bursts extending all the way to 100% utilization. Identical scenarios can be observed for other computer classes, such as desktops, workstations, file servers, and data centers. When observing these utilization traces, it becomes quite obvious that there must be an opportunity to exploit the periods of low activity to reduce energy dissipation.

Slide 10.6
As stated in earlier chapters, the variation in activity moves or changes the optimal E–D curve. In addition, the delay expectation may change as well depending upon operating modes. The optimal operation point hence moves, which means that an energy-efficient design should adapt itself to the changing conditions. Unfortunately, the number of knobs that are available to the designer of the runtime system is restricted. Of the traditional design parameters, only supply and threshold voltages are truly available. Dynamically changing the transistor sizes is not very practical or
Adapting to Variable Workloads

- Goal: Position design in optimal operational point, given required throughput
- Useful dynamic design parameters: $V_{DD}$ and $V_{TH}$
  - Dynamically changing transistor sizes non-trivial
- Variable supply voltage most effective for dynamic power reduction

Adjusting Only the Clock Frequency

- Often used in portable processors
- Only reduces power — leaves energy per operation unchanged
  - Does not save battery life

Slide 10.7
Consider, for instance, the case of the microprocessor embedded in a laptop computer. Assume that the computational tasks can be divided into high-performance tasks with short latency requirements, and background tasks, where the latency is not that important. A processor that runs at a fixed frequency and voltage executes both types of tasks in the same way — this means that the high-performance task is executed within specifications (as shown by the dotted lines), whereas the low-end task is performed way too fast. Executing the latter slower would still meet specifications, and offers the opportunity for power and energy savings.

One approach that was adopted by the mobile-computing industry early on is to turn down the clock frequency when the computer is operating on battery power. Lowering the clock frequency reduces the power dissipation ($P = CV^2f$) proportional to the frequency reduction (assuming that leakage is not a factor). However, it comes with two disadvantages:

- The reduced-frequency processor does fine with the high-latency tasks, but fails to meet the specifications for the high-performance functions;
- Though it scales power, this approach does not change the energy per operation. Hence, the amount of work that can be performed on a single battery charge remains the same.
The first concern can be addressed by changing the frequency dynamically in response to the presented workload. Dynamic Frequency Scaling (or DFS) makes sure that performance requirements are always met (“just in time computing”), but misses out on the energy reduction opportunity.

Slide 10.8
A more effective way of exploiting the reduced workload is to scale clock frequency and supply voltage simultaneously (called dynamic voltage scaling, or DVS). The latter is possible as frequency scaling allows for higher delays, and hence reduced supply voltages. Whereas pure frequency scaling does reduce power linearly, the additional voltage scaling adds a quadratic factor, and reduces not only the average power but also the energy per operation, while meeting all the performance needs.

Slide 10.9
To analyze the effectiveness of DVS, it is worth revisiting the relationship between supply voltage and delay (or clock frequency). Using the \(\alpha\)-delay expression introduced in Chapter 4, and normalizing supply voltage and frequency to their nominal values, an expression between the normalized frequency and required supply voltage can be derived. For long-channel devices, a linear dependency between frequency and voltage is observed. In short-channel transistors, the supply voltage initially scales super-linearly, but the effect saturates for larger reductions in clock frequency.
Slide 10.10
The results of the previous slide can now be used to compute the energy savings resulting from simultaneous supply and frequency scaling. The resulting chart clearly demonstrates that DVS reduces energy per operation super-linearly. Reductions of the clock frequency by factors of 2 and 4 translate into energy savings by factors of 3.8 and 7.4, respectively (in a 90 nm CMOS technology). Scaling only the frequency would have left the energy per operation unchanged – and taking leakage into account, it might even go up!

Slide 10.11
The impact of DVS is even more impressive when considering power dissipation, where an almost third-order reduction can be observed. More precisely, scaling of the clock frequency by factors of 2 and 4 translates into energy savings by factors of 7.8 and 30.6, respectively. Scaling only the clock frequency, leaving the voltage unchanged, would have led to a linear scaling – hence power would have been reduced by factors of 2 and 4, respectively.

Slide 10.12
Having to adjust the supply voltage adaptively and continuously may or may not present a substantial overhead depending upon the system the processor is embedded in. Most microprocessor mother boards include sophisticated voltage regulators that allow for a range of
Using Discrete Voltage Levels

- DVS needs close integration with voltage regulation
- Continuously variable supply voltage not always available

Dithering supply voltage between discrete levels approximates continuous scaling

Example:
- Operate 50% of time at $V_{DDmin}$ and 50% at $V_{DDmin}/2$
- Reduces $e$ to 0.625 for $f=0.74$
- Continuous DVS would yield $e = 0.5$

[Ref: V. Gutnik, VLSI'96]

Different discrete-voltage operation points. The percentage of time spent at each voltage determines the exact operation point. Adding more discrete supply voltages allows for a closer approximation of the continuous DVS curve. This approach is often called voltage hopping or voltage dithering.

For example, if only one extra supply (at $V_{DD}/2$) is available in addition to the nominal supply, spending equal time at both supplies reduces the energy by a factor of 1.6 (instead of the factor 2 that would have resulted from continuous scaling).

Challenge: Estimating the Workload

- Adjusting supply voltage is not instantaneous and may take multiple clock cycles
- Efficiency of DVS a strong function of accuracy in workload estimation
- Depending upon type of workload(s), their predictability, and dynamism
  - Stream-based computation
  - General-purpose multi-processing

or down — and that some energy overhead is incurred in changing the rails.

Hence, measuring and predicting the workload accurately is important. Misestimations can substantially reduce the efficiency of the DVS approach. How to perform workload estimation depends upon the application at hand.
When it is close to full, the processor should speed up; when it empties, the processor can slow down. An output FIFO then translates the variable processing rate into the periodic signal that may be required by the playback device or the communication channel.

Buffer utilization is only one measure of the workload. Its disadvantage is that it comes with extra latency. More sophisticated estimators can be envisioned. For instance, many signal processing and communication algorithms allow for the construction of simple and quick estimators of the computational effort needed. The outcome of these can then be used to control the voltage-frequency loop.

Slide 10.14

Stream processing is a particular class of applications where the workload estimation is relatively straightforward. The video-compression example of Slide 10.4 belongs to this class, and so do audio and voice compression, synthesis, and recognition. In stream processing, new samples are presented at a periodic rate. When buffering the incoming samples into a FIFO, the utilization of the FIFO is a direct measure of the presented workload.

Slide 10.15

The effectiveness of this approach is shown for the case of an MPEG-4 encoder. Each time a new frame arrives, the “scheduler” estimates the amount of work to be performed before the next milestone, and the voltage is adjusted accordingly. In this particular example, the designers choose to use voltage dithering. Analysis showed that just two discrete voltages were sufficient to reap most of the benefits. A power reduction by a factor of 10 was obtained.
Slide 10.16
Another challenge of the DVS approach is how to translate a given “request for performance” into voltage, and subsequently into frequency. One option is to use a self-timed approach (as was adopted in the first ever published DVS paper by Nielsen in 1994). This effectively eliminates the voltage-to-frequency translation step. The performance-to-voltage translation is performed by the closed control loop. Important design choices still need to be made: what voltage steps to apply in response to performance requests, and how fast to respond.

In the synchronous approach, the voltage-to-frequency translation can also be achieved dynamically using a closed-loop approach as well. A dummy delay line, mimicking the worst-case critical path, translates voltage into delay (and frequency).

A third approach is to model the voltage–frequency relationship as a set of equations, or as a set of empirical parameters stored in a look-up table. The latter can be obtained by simulation or from measurements when the chip is started up. To account for the impact of variations caused by temperature changes or device aging, the measurements can be repeated on a periodic base. Table look-up can also be used to translate computational requirements into frequency needs.

Slide 10.17
The closed-loop approach to set the voltage and the frequency is illustrated in this Figure. The difference between the desired and actual clock frequencies is translated into a control signal for the DC–DC converter (after being filtered to avoid rapid fluctuations). The voltage is translated into a clock frequency by the VCO, which includes a replica of the critical path to ensure that all the timing constraints in the processor or computational module are met.
Anyone familiar with phase-locked loops (PLLs) recognizes this scheme. It is indeed very similar to the PLLs commonly used in today’s microprocessors and SoCs to set the clock phase and frequency. The only difference here is that the loop sets the supply voltage as well.

**Table Look-up Frequency–Voltage Translation**

- **User Logic**
  - Calibration Unit (Delay analysis)
  - Frequency-to-Voltage Translation Table (f–V Table)
- **Control**
- **Temp sensor**
- **Vcc**

- **DC–DC Converter**

- **Reference Clock (Ref Clk)**

- **Delay measurements for different voltages obtained from actual module**
- **or using array of ring oscillators**
- **Inverse function (F–V) stored in look-up table, taking into account logic structure**
- **Can compensate for temperature variations**

[Ref: H. Okano, VLSI'06]

**Slide 10.18**

Although the replica critical path approach is an effective and simple way of relating supply voltage and clock frequency, it faces some important challenges in the deep-submicron age. With process variations causing different timing behavior at different locations on the chip, a single replica circuit may not be representative of what transpires on the die. One option is to combine the results of many distributed replica circuits, but this rapidly becomes complex.

An alternative approach is to calibrate the design at start-up time and record the voltage–frequency relationships (compared to an accurate reference clock) in a table. (It is also possible to create the table in advance using simulations.) One possible implementation of this calibration procedure is to apply a sequence of voltages to the logic module and measure the resulting delay. The inverse function, stored in a look-up table, can then be used to translate the requested frequency into the corresponding supply voltage. This approach can accommodate the impact of temperature changes, as the delay–temperature relationship is known or can be determined by simulation in advance. This information can be used to recalibrate the look-up table. A conceptual diagram of a table-based system is shown in this slide.

Another calibration option is to use an array of ring oscillators of different sizes, effectively measuring the actual process parameters. This process information can then be translated into a voltage using a $P–V$ (process–voltage) table, which is obtained in advanced using simulation. This approach, which was proposed in [Okano, VLSI'06], has the advantage that it fully orthogonalizes design- and process-dependent factors.

**Slide 10.19**

All the above considerations are equally valid for general-purpose processors. The major difference is that the workload-to-voltage (or frequency) translation is typically performed in software. Actually, it becomes an operating-system function, the task of which is to translate a set of computational deadlines into a schedule that meets the timing requirements. The processor frequency is just one of the extra knobs that can be added to the set the OS has at hand.

A simple way of estimating the desired clock frequency is to divide the expected number of cycles needed for the completion of the task(s) by the allotted time. For the example of the MPEG
Slide 10.20

The effectiveness of DVS in general-purpose processing is by no small means determined by the quality of the voltage-scheduling algorithm. Task scheduling has been studied intensively in a number of fields such as operations research, real-time operating systems, and high-level synthesis. Much can be learned from what has been developed in those fields.

The maximum savings would be obtained by a so-called Oracle scheduler, which has perfect foreknowledge of the future (as well as all the costs and overheads incurred by a change in voltage), and hence can determine the perfect voltage for every task at hand. The quality of any other scheduling algorithm can be measured by how close it gets to the Oracle schedule.

The worst-performing scheduler is the “ASAP” approach. This puts the processor in idle mode (and associated voltage) when there is no activity, and ramps to the maximum voltage whenever a computation has to be performed.

Most practical scheduling algorithms rely on a number of heuristics to determine the order of task execution as well as to select the design parameters. An example of such is the “zero” algorithm that was proposed in [Pering99].

From the results presented in the table, a number of interesting observations can be drawn.
• The savings that can be obtained by DVS and voltage scheduling are strongly dependent upon the applications at hand. The largest savings occur for applications that do not stress the processor, such as user-interface interactions or audio processing. On the other hand, only small gains are made for demanding applications such as video (de)compression. This is why it is worthwhile to farm these applications out to co-processors, as discussed in Chapter 5.
• Investing in a good scheduling algorithm is definitely worthwhile. Getting within a few percentiles from the Oracle scheduler is possible.

**Slide 10.21**
The impact of using different scheduling algorithms is illustrated in this slide for a “bursty” user-interface application. The supply voltage (and hence energy) levels as obtained by the “ASAP” and “zero” algorithms are compared. The latter raises the supply voltage only rarely above the minimum, does so only for latency-sensitive tasks, and never needs the maximum voltage.

**Slide 10.22**
The voltage- and frequency-setting loop for general-purpose processors pretty much follows the scheme detailed in Slide 10.17. The actual clock frequency $F_{CLK}$ is translated into a digital number by a counter–latch combination (sampled at a frequency of 1 MHz, in this example). The result is compared with the desired frequency, as set by the operating system. The goal of the feedback loop is to get the error frequency $F_{ERR}$ as close as possible to zero.

After filtering, the error signal is used to drive the DC–DC converter (which is an inductive buck converter in this particular case). The supply voltage is translated into the clock frequency $F_{CLK}$ with the aid of a ring oscillator, which matches the critical path of the processor [Burd’00].
0.54 μJ/instruction, or anything in between. If the duty cycle is low, which is often the case in embedded application processors, DVS creates a high-performance processor with a very low average energy per operation. The operational point just moves back and forth on the energy–delay (performance) curve.

### Examples of DVS-Enabled Microprocessors

- **Early Research Prototypes**
  - Toshiba MIPS 3900: 1.3–1.9 V, 10–40 MHz [Kuroda98]
  - Berkeley ARM8: 1.2–3.9 V, 6–85 MIPS, 0.54–5.6 mW/MIPS [Burd00]
- **Xscale:** 180 nm 1.8 V bulk-CMOS
  - 0.7–1.75 V, 200–1000 MHz, 55–1500 mW (typ)
  - Max. Energy Efficiency: ~23 MIPS/mW
- **PowerPC:** 180 nm 1.8 V bulk-CMOS
  - 0.9–1.95 V, 11–380 MHz, 53–500 mW (typ)
  - Max. Energy Efficiency: ~11 MIPS/mW
- **Crusoe:** 130 nm 1.5 V bulk-CMOS
  - 0.8–1.3 V, 300–1000 MHz, 0.85–7.5 W (peak)
- **Pentium M:** 130 nm 1.5 V bulk-CMOS
  - 0.95–1.5 V, 600–1600 MHz, 4.2–31 W (peak)
- **Extended to embedded processors** (ARM, Freescale, TI, Fujitsu, NEC, etc.)

### Slide 10.23

Instead of representing a single operational point, a DVS processor moves dynamically in the energy–delay space. This is illustrated quite nicely with the graph shown in this slide, which plots the operational performance versus the energy, for one of the first DVS processors, published at ISSCC in 2000. Implemented in a 600 nm technology(!), the same processor can implement either an 85 MIPS ARM processor operating at 6.5 μJ/instruction or a 6 MIPS processor at

### Slide 10.24

DVS has progressed immensely since its introduction by the research community. Today, a wide range of embedded, DSP, and notebook processors have embraced the concept. In typical applications, the energy per instruction can vary by as much as a factor of 10.

### Slide 10.25

Although dynamic voltage scaling seems to be a no-brainer for a number of applications (if of course, continuous or discrete supplies can readily be made available), there existed a lot of resistance against its adoption in the early days. The main concerns were related to how it would be possible to guarantee that timing conditions and signal integrity are met under changing conditions. It is already a major challenge to verify that a processor functions correctly for a single
DVS Challenge: Verification

- Functional verification
  - Circuit design constraints
- Timing verification
  - Circuit delay variation
- Power distribution integrity
  - Noise margin reduction
  - Delay sensitivities (local power grid)

Need to verify at every voltage operation point?

All these questions are very pertinent. Fortunately, a number of key properties make the verification task a lot simpler than what would be expected.

Design for Dynamically Varying $V_{DD}$

- Logic needs to be functional under varying $V_{DD}$
  - Careful choice of logic styles is important (static versus dynamic, tri-state busses, memory cells, sense amplifiers)
- Also: need to determine max $|dV_{DD}/dt|$

supply voltage – a task that is getting more complicated with the increasing influence of process variations. Imagine now what it takes if the supply voltage is varied dynamically. Must one check correct functionality at every supply voltage within the operation range? What about the transient conditions while the voltage is being ramped? Must one halt the processor during that time, or can it keep on running?

Slide 10.26
Consider first the issue of ensuring functionality. It is important that the circuit does not fail as a result of the supply voltage changes. This depends upon a number of factors such as the logic style used, or the type of memory cell chosen. Most important is how fast the supply voltage is ramped during transitions.
Static CMOS Logic

Static CMOS operates robustly with varying $V_{DD}$

Slide 10.27
Let us consider the case of complementary CMOS logic. A positive property of this most popular logic style is that the output is always connected to either GND or $V_{DD}$ through a resistive path (during a transition it may temporarily be connected to both). If the output is high and the supply voltage changes, the output of the gate just tracks that change with a short delay owing to the $RC$ time constant. Hence, the functionality of the logic is by no means impacted by DVS. The same is true for a static SRAM cell. In fact, static circuits continue to operate reliably even while the supply voltage is changing.

Dynamic Logic

- Sets strong upper limit on $|dV_{DD}/dt|
- Cannot gate clock in evaluation state
- Tri-state busses fail similarly $\rightarrow$ Use hold circuit

Slide 10.28
This is not the case for dynamic circuits. During evaluation, the “storage” node of the circuit may be at high impedance, and disconnected from the supply network. Ramping the supply voltage during that time period can lead to a couple of failure modes:

- When the supply voltage rises during evaluation, the “high” signal on the storage node drops below the new supply voltage. If the change is large enough ($> V_{TH,PMOS}$), it may be considered a logic-low by the connecting gate.
- On the other hand, when the supply voltage is ramped down, the stored node voltage rises above the supply voltage, and may cause the onset of latch-up, if the difference is larger than the $V_{be}$ of the parasitic bipolar transistor.

These failure mechanisms can be avoided by either keeping the supply voltage constant during evaluation, or by ramping the rails slowly enough that the bounds, defined above, are not exceeded.

High-impedance tri-state busses should be avoided for the same reason.
**Slide 10.29**
The simulated response of a CMOS ring oscillator, shown in this slide, amply serves to validate our argument that static CMOS keeps performing correctly while the voltage is ramped. The plot shows how the output clock signal $f_{\text{clk}}$ keeps on rising while the supply voltage increases.

**Slide 10.30**
Even if a circuit works correctly at one voltage from a timing perspective, this by no means guarantees that it also does so at another one. The relative delays from modules in different logic styles may change owing to the voltage scaling. If so, it may be necessary to check the timing at every supply voltage in the operation range. To evaluate what transpires during voltage scaling, the relative delay (normalized to the delay of a ring oscillator) versus supply voltage is plotted for four typical circuit elements. These include inverter chains, of which the loads are dominated by gate, interconnect, and diffusion capacitance (as each of these has a different voltage dependence). To model paths dominated by stacked devices, a fourth chain consisting of 4 PMOS and 4 NMOS transistors in series is analyzed as well. The relative delay of all four circuits is at a maximum at only the lowest or highest operating voltages, and is either monotonically falling or rising in between. This means that it is sufficient to ensure timing compliance at the extreme ends of the supply voltage range to guarantee compliance everywhere in between. This substantially reduces the timing verification effort.
Note: it may be possible to create a relative-delay curve with a minimum or a maximum occurring in-between the end points, by combining circuits of the different types. However, because the gate-capacitance-dominated delay curve is convex, whereas the others are concave, the combination typically results in a rather flat curve, and the observation above pretty much still holds.

**Slide 10.31**

Another concern is the effect of supply bounce as it may induce timing variations and potential violations. We are not concerned about global supply voltage changes as they affect all timing paths equally and the clock period is adjusted as well – remember that the clock frequency is derived from the supply voltage in a DVS system.

Localized supply variations, however, may only affect the critical paths, and not the clock generator, and can lead to timing violations if the local supply drop is sufficiently large. As such, careful attention has to be paid to the local supply routing. As always, a certain percentage of the timing budget must be set aside to accommodate the impact of supply bounce. However, the question again arises as to the voltage at which the impact of supply noise is the largest and whether we should check it for the complete range.

The sensitivity of delay with respect to $V_{DD}$ can be quantified analytically, and the normalized result is plotted as a function of $V_{DD}$ in this slide. For a submicron CMOS process, the delay sensitivity peaks at approximately $2V_{TH}$. Thus, in the design of the local power grid, we only need to ensure that the resistive (inductive) voltage drop of the power distribution grid meets the design margins for one single supply voltage (i.e., $2V_{TH}$). This is sufficient to guarantee that they are also met at all other voltages.

All in all, though the DVS approach undoubtedly increases the verification task, the extra effort is bounded. In fact, one may even argue that the adaptive closed loop actually simplifies the task somewhat as some process variations are automatically adjusted for.

**Slide 10.32**

So far, we have only considered the dynamic adaptation of the supply voltage. In line with our discussions on design-time optimization, it seems only natural to consider adjusting the threshold voltages at runtime as well. This approach, called *Adaptive Body Biasing* or ABB, is especially appealing in light of the increasing impact of static power dissipation. Raising the thresholds when
Adapative Body Biasing (ABB)

- Similar to DVS, transistor thresholds can be varied dynamically during operation using body biasing
- Extension of DBB approach considered for standby leakage management
- Motivation:
  - Extends dynamic E–D optimization scope (as a function of activity)
  - Helps to manipulate and control leakage
  - Helps to manage process and environmental variability (especially $V_{TH}$ variations)
  - Is becoming especially important for low $V_{DD}/V_{TH}$ ratios

the activity is low (and the clock period high), and lowering them when the activity is high and the clock period short, seems to be a perfect alternative or complement to the DVS approach. It should be apparent that ABB is the runtime equivalent of the Dynamic Body Biasing (DBB) approach, introduced in Chapter 8 to address standby leakage.

In addition to dynamically adjusting the static power, ABB can help to compensate for some of the effects introduced by static or dynamic threshold variations—caused by manufacturing imperfections, temperature variations, aging effects, or all of the above. In fact, if well-executed, threshold variations can be all but eliminated.

![Threshold Variability and Performance](image)

Slide 10.33
As can be observed, variations in thresholds may cause the performance of a module to vary substantially. This effect is more pronounced when the supply voltage is scaled down and the $V_{DD}/V_{TH}$ ratio reduced. Though a 50 mV change in threshold causes a delay change of only 13% at a supply voltage of 1 V (for a 90 nm CMOS technology), it results in a 55% change when the supply is reduced to 0.45 V.

Slide 10.34
The idea of using ABB to address process variations was already introduced in 1994 [Kobayashi94] in a scheme called SATS (self-adjusting threshold voltage scheme). An on-chip leakage sensor amplifies the leakage current (the resistive divider biases the NMOS transistor for maximum gain). When the leakage current exceeds a preset threshold, the well bias generation circuit is turned on, and the reverse bias is increased by lowering the well voltage. The same bias is used for all NMOS
transistors on the chip. Though the circuit shown in the slide addresses the threshold adjustment of the NMOS transistors, the thresholds of the PMOS devices also can be controlled in a similar way.

Note that the overall goal of the SAT scheme is to set leakage to a specific value; that is, the transistor thresholds are set to the lowest possible value that still meets the power specifications.

---

**Slide 10.35**

The effectiveness of the SATS is quite apparent from the measured results shown in this chart. Even with the raw threshold varying by as much as 300 mV, the control loop keeps the actual threshold within a 50 mV range.

---

**Slide 10.36**

This slide features a more recent study of the potential of adaptive body biasing. A test chip implemented by a group of researchers at Intel in a 150 nm CMOS technology [Tschanz02] features 21 autonomous body-biasing modules. The idea is to explore how ABB can be exploited to deal not only with inter-die, but also with intra-die variations. Both the reverse and forward body biasing options are available. Each sub-site contains a replica of the critical path of the circuit under test (CUT), a phase detector (PD) comparing the critical path delay with the desired clock period, and a phase-to-bias converter consisting of a counter, a D/A converter, and an op-amp driver. Only PMOS threshold control is implemented in this
Adaptive Body Bias — Experiment

![Image](image.png)

© IEEE 2002

<table>
<thead>
<tr>
<th>Technology</th>
<th>150 nm CMOS</th>
</tr>
</thead>
<tbody>
<tr>
<td># of sub-sites per die</td>
<td>21</td>
</tr>
<tr>
<td>Sub-site size</td>
<td>1.6 x 0.24 mm</td>
</tr>
<tr>
<td>Body bias range</td>
<td>0.5 V FBB to 0.5 V RBB</td>
</tr>
<tr>
<td>Bias resolution</td>
<td>32 mV</td>
</tr>
</tbody>
</table>

[Ref. J. Tschans, ISSCC'02]

The economic impact of applying ABB should not be ignored either. In the microprocessor world, it is common to sort manufactured dies into frequency bins based on the measured performance (in addition, all of the accepted dies should meet both functionality and maximum power requirements). Without ABB, a majority of the dies ends up in the not-so-lucrative low-frequency bin, whereas a large fraction does not meet specifications at all. The application of per-die and (even more) within-die ABB manages to move a large majority to the high-frequency bin, while pushing parametric yield close to 100%.

From this, it becomes apparent that adaptively tuning a design is a powerful tool of the designer in the nanometer era.
Slide 10.38
ABB is even more effective in circuits operating at low supply voltages and low \( V_{DD}/V_{TH} \) ratios. Under those conditions, a small variation in the threshold voltage can cause either a large performance penalty or a major increase in energy consumption. This is illustrated by the distance between the “nominal” and “worst-case” E–D curves of a complex arithmetic logic function, implemented in a 130 nm CMOS technology.

The supply voltage is variable and ranges between 200 and 500 mV, whereas the threshold voltage is kept constant. Observe that the delay is plotted on a logarithmic scale.

A substantial improvement is made when we allow the threshold voltages to be tuned. One option is to simultaneously modify all threshold voltages of a module by adjusting the well voltage. Even then, the worst-case scenario still imposes a performance penalty of a factor of at least two over the nominal case. This difference is virtually eliminated if the granularity of threshold adjustment is reduced – e.g., allowing different body bias values for every logical path. Overall, introducing “adaptive tuning” allows a performance improvement by a factor of 12 over the worst-case un-tuned scenario, while keeping energy constant.

Slide 10.39
At this point, it is only a small step to consider the advantages of simultaneously applying DVS and ABB. Whereas DVS mainly addresses the dynamic power dissipation, ABB serves to set the passive power to the appropriate level. This combined action should lead to E–D curves that are superior to those obtained by applying the techniques separately.

An example of a circuit incarnation that adjusts both \( V_{DD} \) and \( V_{TH} \) is shown on this slide [Miyazaki02]. The requested workload is translated into a desired supply
voltage using the table look-up approach. Given the selected $V_{DD}$, a replica of the critical path is used to set the well voltages for NMOS and PMOS transistors so that the requested clock frequency is met. This approach obviously assumes that the replica path shows the same variation behavior as the actual processor.

**Slide 10.40**

Actual measurements for the circuit of Slide 10.39 indeed show the expected improvements. Compared to DVS only, adding ABB improves the average performance of the circuit substantially for the same power level (and vice versa).

**Slide 10.41**

Although these performance improvements are quite impressive, one may question how effective the combined DVS/ABB approach is in suppressing the effects of (threshold) variations. The chart on the left side of the slide plots the measured clock frequency and power numbers for the same circuit as collected from a large number of dies (over different wafers). For these measurements, supply voltages were fixed to the nominal value, and no body biasing was applied. Though the measurements show a very broad and wide distribution (20% in both clock frequency and power), a general trend can be observed – that is, slower circuits consume less power (this obviously is not a surprise!).

With the introduction of DVS and ABB, circuits that do not meet the performance or power specification are adjusted and brought within the acceptable bounds (with the exception of some circuits that cannot be corrected within the acceptable range of supply and bias voltages, and hence should be considered faulty). The resulting distribution is plotted on the right, which indicates that
dynamic adaptation and tuning is indeed a very effective means of addressing the impact of technology and device variations.

One very important caveat should be injected here: just when device variations are becoming a crucial design concern, one of the most effective means of combating them – that is body biasing – is losing its effectiveness. As we had already indicated in Chapter 2, the high doping levels used in sub-100 nm technologies reduce the body-effect factor: at 65 nm and below, ABB may be barely worth the effort (if at all). This is quite unfortunate, and is hopefully only temporary. The introduction of novel devices, such as dual-gate transistors, may restore this controllability at or around the 32 nm technology node.

**Slide 10.42**

Another important and general observation is worth making. The DVS and ABB schemes, presented in the previous slides, are great examples of a new class of circuits (called self-adaptive) that deal with variations (be it caused by changes in activity, manufacturing, or the environment) by using a closed feedback loop. Online sensors measure a set of indicative parameters such as leakage, delay, temperature, and activity. The resulting information is then used to set the value of design parameters such as the supply voltage and the body bias. In even more advanced schemes, functions might even be moved to other processing elements if performance requirements cannot be met.

The idea is definitely not new. In the 1990s, high-performance processors started to incorporate temperature sensors to detect over-heating conditions and to throttle the clock frequency when the chip got too hot. The difference is that today’s self-adaptive circuits (as adopted in high-end products) are a lot more sophisticated, use a broad range of sensors, and control a wide range of parameters.

**Slide 10.43**

Although adaptive techniques go a long way in dealing with runtime variability, ultimately their effectiveness is limited by the “worst-case” conditions. These may be the voltage at which the timing constraints of a critical path cannot be met or when a memory cell fails. In a traditional-design approach, this is where the voltage scaling ends. However, on closer inspection, one realizes that these worst-case conditions occur only rarely. Hence, if we can cheaply detect the occurrence of such a condition and correct the resulting error when it occurs, we could over-scale the voltage, further reducing the energy dissipation.

Let us, for instance, consider the case of an SRAM memory. As we had discussed in Chapter 9, the minimum operational voltage of an SRAM cell (the DRV) varies cell-by-cell. Fortunately, the
Aggressive Deployment (AD)

- Also known as “Better-than-worst-case (BTWC) design”
- Observation:
  - Current designs target worst-case conditions, which are rarely encountered in actual operation
- Remedy:
  - Operate circuits at lower voltage levels than allowed by worst case, and deal with the occasional errors in other ways

Example:
Operate memory at voltages lower than that allowed by worst case, and deal with the occasional errors through error correction

Distribution ensures that error rate is low

Aggressive Deployment – Concepts

- Probability of hitting tail of distribution at any time is small
  - Function of critical-path distribution, input vectors, and process variations
- Worst-case design expensive from energy perspective
  - Supply voltage set to worst case (+ margins)
- Aggressive deployments scales supply voltage below worst-case value
  - “Better-than-worst-case” design strategy
  - Uses error detection and correction techniques to handle rare failures

Hence, knowing the distribution of the critical parameters is important.
- The worst-case scenario leaves a large number of crumbs on the table. All circuitry in a module consumes way too much energy just because of a small number of outliers.
- Hence it pays to let some errors occur by over-scaling, and condone the small overhead of error detection and correction.

Slide 10.44

The basic concepts upon which this “better-than-worst-case” (BTWC) (first coined as such by Todd Austin) design is built are as follows:

- Over-scaling of the supply voltage leads only to rare errors, not to catastrophic breakdown. In the latter case, the overhead of dealing with the errors would dominate the savings.
Slide 10.45
Like DVS and ABB, BTWC (very often also called aggressive deployment, or AD) relies on the presence of a feedback loop, positioning the system at its optimal operation point from a performance/energy perspective. A BTWC system consists of the following elements:

- A mechanism for setting the supply voltage based on the understanding of the trade-off between introducing errors and correcting them. In one way or another, this control mechanism should be aware of the error distribution (either by simulation in advance, or by adaptive learning).

- An error-detection mechanism – As this function is running continuously, its energy overhead should be small. Coming up with efficient error-detection approaches is the main challenge in the conception of BTWC systems.

- An error-correction strategy – As errors are expected to be rare, it is ok to spend some effort in correcting them. The correction mechanisms can vary substantially, and depend upon the application area as well as the layer in the abstraction chain where the correction is performed.

It is important to realize that the BTWC approach is very generic, and can be applied at many layers of the design abstraction chain (circuit, architecture, system) and for a broad range of application spaces, some of which are briefly discussed in the following slides.
Once we start lowering the supply voltages, some timing paths may not be met and errors start to appear. In the FPGA prototype, the first errors occur at 1.54 V. Observe that the y-scale of this plot is logarithmic. If we would allow for a 1.3% error rate (which means that one out of 75 samples is wrong), the supply voltage can be scaled all the way down to 1.36 V. This translates into a power reduction of 35%.

**Slide 10.46**
As a first example, let us consider what happens when we lower the supply voltage of a logical module such as a multiplier, which typically has a wide distribution of timing paths. In a traditional design, the minimum supply voltage is set by the worst-case timing path with an extra voltage margin added for safety. The example on the slide shows the results for an 18×18 multiplier, implemented on an FPGA. Including the safety margin, the minimal operational voltage equals 1.69 V.

**Slide 10.47**
It is worth observing that error rate and the shape of the error histogram are functions of the data patterns that are applied. For the example of a multiplier, random data patterns tend to trigger the worst-case paths more often than the correlated data patterns that commonly occur in signal-processing applications. The same holds for many other computational functions, as is illustrated in this slide for a Kogge–Stone adder. When applying data patterns from applications such as *bzzip* or *annmp*, the voltage can be scaled down by an extra 200 mV for the same error rate.

Hence, to be effective, the voltage-setting module must somehow be aware of the voltage-to-error function. As for DVS, this information can be obtained by simulation or during a training period and can be stored in a table.
the other hand, the main clock arrives too early and a path has not stabilized yet, main latch and shadow latch will capture different values. A sole XOR is sufficient to detect the error.

The above description is somewhat over-simplifying, and some other issues need to be addressed for this approach to work. For instance, no shadow latches should be placed on “short paths” as this may cause the shadow latch to catch the next data wave. In other words, checking the set-up and hold-time constraints becomes more complicated. Also, the first latch may get stuck in a metastable state, leading to faulty or undecided error conditions. Extra circuitry can be added to get around this problem. For more detailed information, we refer the interested reader to [D. Ernst, MICRO’03].

Upon detection of an error, a number of strategies can be invoked for correction (depending upon the application space). For instance, since the original RAZOR targets microprocessors, it passes the correction task on to the micro-architectural level.
with some cycle (and energy) overhead, but remember: errors are expected to occur only rarely if the voltage-setting mechanism works correctly.

**Slide 10.50**

The voltage-setting mechanism is a crucial part of any AD scheme. In the RAZOR scheme, the errors per clock cycle occurring in the data path are tallied and integrated. The error rate is used to set the supply voltage adaptations. As mentioned earlier, knowledge of the voltage-error distribution helps to improve the effectiveness of the control loop.

**Slide 10.51**

Given our previous discussion of adaptive optimizations in the delay–energy space, it should come as no surprise that BTWC schemes converge to an optimal supply voltage that minimizes the energy per operation. Reducing the supply voltage lowers the energy, but at the same time increases the correction overhead. If the voltage–error relationship is gradual (such as the ones shown for the multiplier and the Kogge–Stone adder), the optimal operational point shows a substantial improvement in energy for a very small performance penalty.
Trade-off curves such as the one shown are typical for any BTWC approach, as will be demonstrated in some of the subsequent slides.

**Caveat:** For aggressive deployment schemes to be effective, it is essential that the voltage–error distribution has a “long tail”. This means that the onset of errors should be gradual once the supply voltage is dropped below its worst-case value. The scheme obviously does not work if a small reduction leads to “catastrophic failures”. Unfortunately, a broad range of the energy-reduction techniques, introduced earlier in this book, tend to create just this type of condition. Design-time techniques, such as the use of multiple supply and threshold voltages as well as transistor sizing, exploit the slack on the non-critical paths to minimize energy dissipation. The net result of this is that a larger percentage of timing paths become critical. Under such conditions, a small voltage reduction can lead to a catastrophic breakdown. This opens the door for an interesting discussion: wouldn’t it be better to forgo the design-time optimizations and let the runtime optimizations do their job – or vice versa? The only way to get a relevant answer to this question is to exploit the systematic system-level design exploration framework, advocated in this book.

---

**Slide 10.52**

Although concepts such as runtime adaptation and BTWC design show great potential, it takes substantial effort to transfer them into producible artifacts. Similar to DVS, AD requires a re-evaluation of the standard design flows and a rethinking of traditional design concepts. As we had mentioned in the previous slide, concepts such as RAZOR require an understanding of how and when a chip breaks. To make the approach more effective, we may even want to rethink accepted design technologies. But the benefits of doing so can be very substantial.
In this slide, the impact of applying the RAZOR concept to an embedded processor of the ARM™ family is shown. An energy reduction of at least 39% over all processors is obtained, whereas average savings are at least 50%. Getting to this point required a major redesign not only of the data path but also of the memory modules. But the results show that that effort is ultimately very rewarding.

Slide 10.53
The RAZOR concept combines error detection at the circuit level with error correction at the micro-architecture level. Many other BTWC strategies can be envisioned. For example, in this slide we show an approach that employs both error detection and correction at the algorithm level.

One interesting property of many signal-processing and communication applications is that the theory community has provided us with simple ways to estimate the approximate outcome of a complex computation (based on the past input stream). The availability of such estimates provides us with a wonderful opportunity to reduce energy through BTWC.

The “Main Block” in the diagram represents some complex energy-intensive algorithm, such as for instant motion compensation for video compression. In normal operation, we assume this block to be error-free. Assume now that we aggressively scale the supply voltage of this block so that errors start to occur. In parallel with the “Main Block”, a simple estimator is run which computes the expected outcome of the “Main Block”. Whenever the latter ventures to values far from the prediction, an error condition is flagged (detection), upon which the faulty outcome is replaced by the estimation (correction). This obviously deteriorates the quality of the processor – in signal-processing speak, it reduces the signal-to-noise ratio (SNR). However, if the estimator is good enough, the increase in the noise level is masked by the noise of the input signal or by the added noise of the signal-processing algorithm, and hence barely matters. Also observe that “small errors” (errors that only affect the least-significant bits [LSBs]) may go undetected, which is ok as they only impact the SNR in a minor way.

As for RAZOR, algorithmic BTWC leads to an optimal supply voltage. If the error rate gets higher, the error correcting overhead starts to dominate (in addition, the deterioration in SNR may not be acceptable). For this scheme to work, clearly it is essential that the estimator does not make any errors itself. This requires that the “Estimate Module” be run at the nominal voltage. Since it is supposed to be a simple function, its energy overhead is small.
a reduced sampling rate (through sub-sampling). Only the MSAD is voltage scaled. A pleasant surprise is that in the presence of process variations, the AD version performs better than the original one, from an SNR perspective. It turns out that this is not an exception – techniques that exploit the joined statistics of the application and the process often end up performing better than those that don’t.

Slide 10.54
The effectiveness of algorithmic-level AD is demonstrated with a video compression example, more specifically for its motion estimation block, which is the most compute-intensive function. The main algorithm uses MSAD (main sum of absolute differences), whereas the estimator uses a simpler version called ISR-SAD (Input-sub-sampled replica of sum of absolute differences). The main simplifications used in the estimator are a reduced precision as well as

Slide 10.55
As mentioned, the concepts of runtime adaptation and AD are broad and far-reaching. We have only shown a couple of examples in this chapter. A number of other instantiations of the concept are enumerated on this slide. We also suggest that you consult the March 2004 issue of the IEEE Computer Magazine, which features a variety of BTWC technologies.
### Power Domains (PDs)

**Introduction of multiple voltage domains on single die creates extra challenges:**
- Need for multiple voltage regulators and/or voltage up-down converters
- Reliable distribution of multiple supplies
- Interface circuits between voltage domains
- System-level management of domain modes
  - Trade off gains of changing power modes with overhead of doing so
  - Centralized “power management” very often more effective

or at the data retention level, while other modules are active and require either the full supply or a dynamically varying one. Each chip partition that needs individual power control is called a **power domain** (PD).

The introduction of power domains in the standard design methodology comes with some major challenges. First of all, generating and distributing multiple variable supply voltages with a reasonable efficiency is not trivial. Many of the gains made by varying supply and well voltages could be lost if the voltage conversion, regulation, and distribution is not done efficiently. Another, often forgotten, requirement is that signals crossing power boundaries should be carefully manipulated. Level conversion, though necessary, is not sufficient. For instance, the output signals of an active module should not cause any activity in a connected module in standby; or, vice versa, the grounded output signals of a standby module should not result in any erroneous activity in a connected active block.

The most important challenge however is the **global power management** – that is, deciding what voltages to select for the different partitions, how fast and how often to change supply and well voltages, when to go in standby or sleep mode, etc. In the preceding slides and chapters, we had introduced voltage-setting strategies for individual modules. A distributed approach, in which each module individually chooses its preferred setting at any point in time, can be made to work. Yet, it is often sub-optimal as it lacks awareness of the global state of the system. A centralized **power manager** (PM) often can lead to far more efficient results.

### Slide 10.57

There are a couple of reasons for this. First of all, the PM can examine the global state of the system, and may have knowledge of the past state. It is hence in a better position to predict when a block will become active or inactive, or what the level of activity may be. Furthermore, transferring the state to a centralized module allows a sub-module to go entirely dormant, reducing leakage power. For instance, many sleep strategies often employ timers to set the next wake-up time (unless an input event happens earlier). Keeping the timers running eliminates the possibility of complete power-down of the unit. Hence, transferring the time-keeping to a centralized “scheduler” makes clear sense.

Although many SoCs employ some form of a power management strategy, most often it is constructed ad hoc and after the fact. Hence, a methodological approach such as the one advocated
in this slide is advisable. A coordinated PM contains the following components: a central control module (called event/command dispatcher), and time, power, and clock sub-systems. The latter contain the necessary knowledge about past and future timing events, power- and voltage-setting strategies for the individual modules, and the voltage-clock relationships, respectively. The dispatcher uses the information of the three sub-systems to set a voltage-scheduling strategy for the different power domains on the chip. Inputs from PDs (such as a request to shut down, or a request to set up a channel to another PD), as well as scheduling decisions and power-setting commands are interchanged between the PDs and the PM over a “power network” with standardized interfaces.

In a sense, the PM takes on some of the tasks that typically would be assigned to a scheduler or an operating system (OS) (which is why another often-used name for the PM is the “chip OS”). However, the latter normally runs on an embedded processor, and consequently that processor could never go into standby mode. Dedicating the PM functionality to a specialized processor avoids that problem, with the added benefit that its energy-efficiency is higher as well.

**Managing the Timing**

- **Basic scheduling schemes**
  - Reactive
    - Sleep when not actively processing
    - Wake up in response to a pending event
  - Stochastic
    - Sleep if idle and probably not needed in near future [Simunic’02]
    - Wake up on account of expected event in the near future

- **Metrics**
  - Correctness – PD awake when required to be active
  - Latency – time required to change modes
  - Efficiency – minimum total energy consumption [Liao’02]
    - Minimum idle time – time required for savings in lower-power mode to offset energy spent for switching modes
    - Min. Idle Time = \( \frac{E_{idle}}{P_{ave}} = \frac{E_{switch} - E_{idle,switch}}{P_{idle} - P_{idle}} \)

PhD theses of Tajana Simunic (Stanford) and Mike Sheets (UCB) probably present the most in-depth treatments on the topic so far.
Interfacing Between Power Domains

**Separate internal logic of block from its interfaces**
1. Communicate with other PDs by bundling related signaling into “ports”
   - Communication through a port requires permission (session-based)
   - Permission is obtained through power-control interface
2. Signal wall maintains interface regardless of power mode
   - Can force to a known value (e.g., the non-gated power rail)
   - Can perform level conversion

A structured approach to the construction of PDs can also help to address the challenge of the proper conditioning of signals crossing power domains. Putting a wrapper around every PD supporting only standardized interfaces makes the task of composing a complex SoC containing many PDs a lot simpler. For instance, the interface of each PD should support a port to communicate with the PM through the “Power Network” and a number of signaling ports to connect to other PDs. The signaling ports can contain a number of features such as level conversion or signal conditioning, as shown in the slide.

Example: PDs in Sensor Network Processor

- **2.7x2.7 mm² (130 nm CMOS)**
  - Clock Rates: 8–60 KHz
  - Supply: 0.3–1 V
  - Leakage Power: 53 μW
  - Average Power: 150 μW
  - Peak Power: 5 mW

An example of a structured power management approach is shown on this slide. This integrated protocol and application processor for wireless sensor networks combines a broad range of functions such as the baseband, link, media-access and network-level processing of the wireless network, node locationing, as well as application-level processing. These tasks exhibit vastly different execution requirements — some of them are implemented in software on the embedded 8051 micro-controller, whereas the others are implemented as dedicated hardware modules — as well as dissimilar schedules. It is rare for all functions to execute simultaneously. To minimize standby power (which is absolutely essential for this low duty cycle application), an integrated power manager assumes that any module is in power-down mode by default. Modules transition to active mode as a result of either timer events (all timers are incorporated in the PM), or events at their input ports. For a module in standby, the supply voltage is ramped down either to GND if there is no state, or to a data retention
voltage of 300 mV. The latter is the case for the embedded micro-controller, whose state is retained in between active modes. To minimize overhead, the retention voltage is generated by an on-chip voltage converter. When the chip is in its deepest sleep mode, only the PM running at an 80 kHz clock frequency is still active.

The logic analyzer traces show how all modules are in standby mode by default. Power is mostly consumed during a periodic RX cycle, when the node is listening for incoming traffic, or during a longer TX cycle. Modules servicing the different layers of the protocol stack are only fired up when needed. For instance, it is possible to forward a packet without waking up the micro-controller.

---

**Integrated Switched-Capacitor Voltage Converter**

**Output voltage ripple function of \( R_{\text{load}} \) and \( f_{\text{clk}} \)**

[Ref: H. Qin. ISQED'04]

---

**Slide 10.61**

For low-power applications, such as wireless sensor networks, using off-the-shelf components to generate the various voltages that are needed on the chip turns out to be very inefficient. Most commercial voltage regulators are optimized for high-power applications drawing Amperes of current. When operated at mW levels, their efficiency drops to the single-digit percentage level (or even lower). Hence, integrating the regulators and converters on-chip is an attractive solution. The fact that the current demands for these converters are very low helps substantially in that respect. An additional benefit of the integrated approach is that the operational parameters of the converter can be adapted to the current demand, maintaining a high level of efficiency over the complete operation range.

The “switched-capacitor” (SC) converter, shown in this slide, works very well at low current levels, and can be easily integrated on a chip together with the active circuitry (such as in the case of the sensor-network processor of Slide 10.60). No special demands are placed on the technology. The ripple on the output voltage is determined by the current drawn (represented by the load resistor \( R_{\text{load}} \)), the total capacitance in the converter, and the clocking frequency. During the standby mode, the load resistance is large, which means that the clock frequency of the converter can be reduced substantially while keeping the voltage ripple constant. Hence, high levels of efficiency can be maintained for both active and standby modes. The only disadvantage is that the capacitors composing SC converters consume a substantial amount of silicon area. This makes their use prohibitive for applications that draw a substantial amount of current. Advanced packaging technologies can help to offset some of these concerns.

---

**Slide 10.62**

Using the SC-converter concept, it is possible to create a fully integrated power train for wireless sensor applications. As was mentioned in the introduction chapter (Chapter 1), distributed sensor
network nodes strive to harvest their energy from the environment to ensure operational longevity. Depending upon the energy source, rectification may be necessary. The scavenged energy is temporarily stored on either a rechargeable battery or a supercapacitor to balance supply and demand times. The sensor node itself requires a variety of voltages. Sensors, for example, tend to require higher operational voltages than digital or mixed-signal hardware. A bank of switched-capacitor converters can be used to provide the necessary voltage levels, all of which need to have the possibility to be ramped down to zero volt or the DRV for standby.

A dedicated integrated power-conversion chip, accomplishing all these functions for a wireless sensor node targeting tire-pressure monitoring applications, is shown in this slide. The IC contains the rectifiers as well as the various level converters. High levels of conversion efficiency are maintained over all operational modes.

Slide 10.63
Unfortunately, SC voltage converters are only effective at low current and power levels (i.e., at the mA and mW range). Most integrated circuits run at substantially higher current levels, and require more intricate voltage regulators and converters. The most efficient ones are based on resonant LC networks (most often called buck converters), where energy is transferred with minimal losses between an inductor and a capacitor at a well-defined switching rate. An example of such a converter is shown on the slide.
LC-based voltage regulators are generally implemented as stand-alone components. There is a very good reason for this: the required values and quality factors of the inductors and capacitors are hard to accomplish on-chip. Hence, the passives are most often implemented as discrete components. For SoCs with multiple power domains and dynamically varying voltage requirements, there are some compelling reasons to strive for a tighter integration of the passives with the active circuitry. Direct integration of the controller circuitry with the load leads to more precise control, higher efficiencies, and increased flexibility.

Though integrating the Ls and Cs directly on the IC may not be feasible, an alternative approach is to implement the passives on a second die (implemented on a substrate of silicon or some other material such as a plastic/glass interposer), which provides high-quality conductors and isolators, but does not require small feature sizes. The dies can then be connected together using advanced packaging strategies. An example of such a combined inductor/capacitor circuit is shown. Capacitance and inductance values in the nF and nH range, respectively, can be realized in this way. The concept of stacking dies in a 3D fashion is gaining rapid acceptance these days – driven mostly by the size constraints of mobile applications. This trend surely plays in favor of closer integration of the power regulation with the load circuitry, and of distributed power generation and conversion.

**Slide 10.64**

The possibility of multidimensional integration may lead to a complete rethinking of how power is distributed for complex SoCs. In the ICs of the past years, the power distribution network consisted of a large grid of connected Copper (or Al) wires, all of which were set to the nominal supply voltage (e.g., 1 V). The concept of power gating has changed this practice a little: instead of being connected directly to the power grid, modules now connect through switches that allow an idle module to be disconnected.

If it becomes possible to integrate voltage converters (transformers) more tightly into the network, a totally new approach may arise. This would resemble the way power is distributed in large scale at the metropolitan and national levels: the main grid is operated at high voltage levels, which helps to reduce the current levels and improves the efficiency. When needed, the power is down-converted to lower levels. In addition to the introduction of transformers, switches also can be introduced at multiple levels of the hierarchy.
Slide 10.65
A graphical representation of the constructions this vision could lead to is shown. Most standard circuitry is implemented on the base chip. Also included on the die is the control circuitry for the power regulators of the various power domains. However, the power grids of the latter are not connected on the die. The “higher levels” of the power distribution network are implemented on an interposer die, which implements a grid of high-quality inductors and capacitors, as well as a high-voltage power grid. The 2.5D integration strategy also allows for non-traditional technologies such as MEMs, or non-digital technologies such as DRAMs, to be tightly integrated with the computational fabric in a compact package.

Note: The term 2.5D integration relates to a three-dimensional IC technology, where individual dies are stacked on top of each other and interconnected using solder bumps or wire bonding. A true three-dimensional integration strategy, on the other hand, supposes that all active and passive devices are realized as a single artifact by constructively creating a stack of many layers deposited on top of one another. Although this ultimately may be the better solution, a large volume of economical and technological issues make the latter approach quite impractical for the time being.

Slide 10.66
In summary, the combination of variations in activity, process, and environmental conditions is leading to fundamental changes in the way ICs and SoCs are being designed and managed. Rather than relying solely on design-time optimizations, contemporary integrated circuits adjust parameters such as the supply and well voltages on the fly, based on observation of parameters such as the workload, leakage, and temperature. In addition, different parameter sets can be applied to individual regions of the chip called power domains.
This design strategy represents a major departure from the methodologies of the past. It challenges the standard design flows – yet does not make them obsolete. Actually, upon further contemplation, we can come to the conclusion that the idea of runtime optimization may make the traditional design strategies more robust in light of the challenges of the nanometer era, at the same time helping to reduce energy substantially.

Literature

Books, Magazines, Theses

Articles

References (cont.)