Energy Management Techniques for SoC Design

Yasuura, Hiroto
System LSI Research Center, Kyushu University

Ishihara, Tohru
System LSI Research Center, Kyushu University

Muroyama, Masanori
System LSI Research Center, Kyushu University

出版情報：Essential Issues in SOC Design: Designing Complex Systems-on-Chip, pp.177-223, 2006-09. Springer
バージョン：
権利関係：

Chapter 6

ENERGY MANAGEMENT TECHNIQUES FOR SOC DESIGN

Hiroto YASUURA, Tohru ISHIHARA, Masanori MUROYAMA
System LSI Research Center, Kyushu University, 3-8-33, Momochihama, Sawara-ku, Fukuoka, 814-0001, JAPAN

Abstract: One of the biggest problems in complicated and high-performance SoC design is management of energy and/or power consumption. In this chapter, we present energy management techniques in system design including HW and SW, SoC architecture and logic design. Dynamic power consumption is the major factor of energy consumption in the current CMOS digital circuits. The dynamic power consumption is affected by supply voltage, load capacitance and switching activity. We present approaches to controlling supply voltage, load capacitance and switching activity dynamically and statically in system architecture and algorithm design levels. We also discuss on the memory architecture for reducing power and energy in HW and SW co-design of SoC. In the future CMOS technology, leakage power consumption becomes dominant, because the threshold voltages are scaled as the transistor size shrinks. We summarize the techniques for reducing leakage power in system architecture design. The contents of the chapter include the following issues; (1) power and energy consumptions in SoC design, (2) tradeoff between energy and performance, (3) tradeoff among energy, QoS (i.e., latency and computational precision), reliability, and flexibility (4) techniques for reducing dynamic power consumption, and (5) leakage power reduction techniques.

Key words: Energy consumption, Power consumption, Reliability, Quality of service, HW and SW co-design

1. INTRODUCTION

In past years, the most serious concerns for the VLSI designer were performance, cost, and reliability. Recently, however, this paradigm has

shifted. More specifically, reducing power and/or energy consumption has become one of the most important themes in SoC design. The driving factors of the paradigm shift include the followings.

- Popularization of portable electronic devices
- Raising demand for reliable and stable computer systems
- Worldwide environmental destruction

One of the biggest factors which motivate the need for low power SoC is the popularization of portable electronics. The typical power consumption for a portable multimedia terminal is around the range of 10-50 [W] when employed chips are not optimized for low-power. Assuming a battery yielding around 65 watt-hours per kilogram is used, the terminal would require unacceptable six kilograms of batteries for ten hours operation between recharges. If we use 500 grams of batteries, the terminal operates only one hour without recharges. Therefore, it is clear that the power consumption has a strong impact on a value of the portable electronic products.

The second need for low power comes from a strong pressure for designers of high-end products to reduce their temperature. In [Black69], Black mentioned that the Mean Time To Failure (MTTF) of aluminum interconnects exponentially decreases as the temperature of a chip increases. Therefore, cooling down the chip temperature is essential for a reliable and stable operation of computer systems. Contemporary performance-optimized microprocessors dissipate as much as 15-50W at 100-200MHz clock rates. The leakage power issue makes this situation worse, because the leakage power increases exponentially as the temperature of the chip increases. In the future, it is expected that a 10 cm^2 microprocessor with 500MHz clock frequency consumes about 300W. The cost for cooling such chips is huge. Consequently, there is a clear advantage to reducing the power consumed in computer systems. Especially for consumer products whose sales are strongly affected by its price, lowering the power is indispensable.

Worldwide environmental destruction drives the strong need for low power electronic devices. Although the power consumption of each electronic device is small (around the range of 10-50W), they are used anywhere and anytime in today's highly information oriented society. Assuming coverage of such electronic devices in the world is 50%, 3.3 billions of people waste 500 billions of watts of power. In addition, rising IT population accelerates this situation. If we reduce the energy consumption of the electronic devices by 10%, we can save 65 mega tons of oil used in gas turbine power plants per a year or can reduce 50 nuclear power units. In 10-20 years from now, we need to come up with innovative solutions which drastically save the energy of the electronic devices with accelerating the growth of IT population.

Recently, many energy reduction techniques at various levels of abstraction, such as at device, circuit, layout, architectural, and software levels are proposed. Regarding the physical design, energy optimization techniques are well studied. However, there is much scope left to study in the system level such as architectural, algorithm, or software level. In this chapter, we present system level energy reduction techniques which might be an essential in SoC design.

The rest of the chapter is organized in the following way. In Section 2, we explain mechanisms of power and energy dissipations in CMOS circuits and summarize basic strategy for reducing power and energy consumptions. Section 3 presents techniques for lowering supply voltage statically or dynamically considering several design tradeoffs. Section 4 presents techniques for reducing switching activity without sacrificing quality of services (QoS). In Section 5, we present techniques for reducing the product of switching activity and load capacitance. Section 6 presents strategies for reducing leakage power and shows several examples in detail. Section 7 summarizes techniques for reducing energy consumption by customizing hardware for the target application. Section 8 concludes this chapter.

2. POWER AND ENERGY CONSUMPTIONS IN SOC

The energy consumption of a system, *E*, can be defined as the summation of both spatial and temporal power consumption of circuits [Weste93] as shown in (1) and (2).

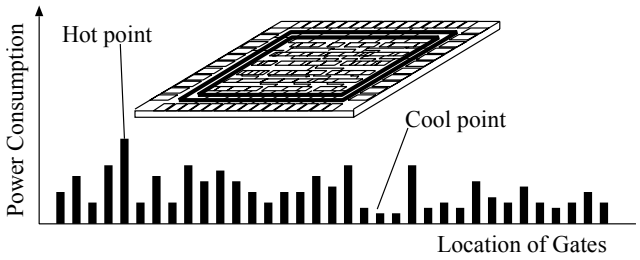


Figure 6-1. Local Power Dissipation

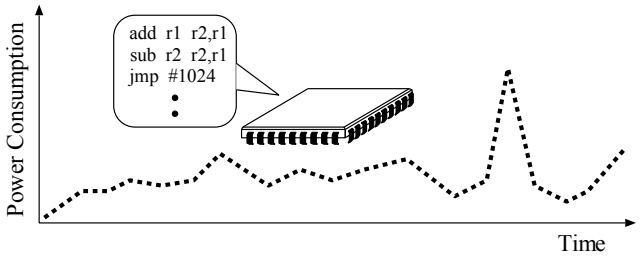


Figure 6-2. Power Dissipation vs. Energy Dissipation

$$P = P_{dynamic} + P_{leak} = \sum_{g \in G} SA(g) \cdot CL(g) \cdot V_{DD}(g)^2 + P_{leak}(g) \qquad (2.1)$$

$$E = \int_0^t P dt \qquad (2.2)$$

P: Power consumption of the target system
$P_{dynamic}$: Dynamic power consumption of the target system
P_{leak}: Leakage power consumption of the target system
$SA(g)$: switching activity of gate g (expected number of 0->1 transitions per second)
$CL(g)$: load capacitance of g
$V_{DD}(g)$: operation voltage of g
t: Execution time of an application program

We treat the energy consumption, E, as an objective function to be optimized, because the energy consumption is close related to the heat and reliability of chips, battery life time of portable devices, and the number of nuclear and gas turbine power stations required. The main approach is detecting a spatial and temporal *hot spot* and reducing the power consumption of the spot. Since the power consumption, P, dynamically changes according to the behavior of the software running on a chip and a location of the logic gate on the chip as shown in figures 6-1 and 6-2, both the software and the hardware should be taken into account for reducing the energy consumption of a SoC chip. As one can see from Equations (2.1) and (2.2), we can reduce the energy consumption of the SoC chip by lowering $SA(g)$, $CL(g)$, $V_{DD}(g)$, $P_{leak}(g)$ and t. However, lowering these parameters sometimes causes an increase of the execution time, a degradation of computational quality, system reliability and design flexibility. The key point of the energy reduction in SoC design is considering design tradeoffs

among energy consumption, performance, computational quality, system reliability and design flexibility. The goal is minimizing the energy consumption under the constraint of performance, computational quality, system reliability and/or design flexibility.

There is a third source of power consumption, short-circuit power, which results from a short-circuit current-path between the power supply and ground during switching. Short-circuit power is projected to be constant around 10% of total power consumption for succeeding technologies [Chatterjee96]. We ignore it throughout this chapter.

There are many techniques proposed for reducing the execution time t, and some of them are very effective for reducing the energy consumption of the SoC chip. In this chapter, however, we do not focus on the techniques which mainly aim to reduce the execution time. Instead, we summarize techniques which consider the execution time as a design constraint. In this chapter, we will make a brief survey on approaches to reducing *SA(g), CL(g)*, $V_{DD}(g)$, and $P_{leak}(g)$ in SoC design. We will also clarify the basic strategy underlying the approaches and show several examples in detail.

3. TECHNIQUES FOR LOWERING OPERATING VOLTAGE

Since energy dissipation is quadratically proportional to supply voltage (see equation (2.1)), lowering the V_{DD} has a strong impact on the energy reduction. However, the following drawbacks should be taken into account;

1. loss of compatibility to external voltage standards,
2. performance degradation, and
3. reliability issues (very low voltage).

3.1 Compatibility of different voltage standards

Whenever one circuit has to drive an input of another circuit operating at a higher supply voltage, a level conversion is needed at the interface. Suppose we have two different voltages, V_{DH} and V_{DL} ($V_{DH} > V_{DL}$). If the output of a circuit operating at V_{DL} is connected directly to the input of a circuit operating at V_{DH}, the static current flows in the input cells operating at V_{DH}, because the PMOS of the input cells cannot be cut-off as shown in Figure 6-3.

In these days, it is common to have level shifting cells in a cell library for accepting multiple signal levels on a chip. Usami et al. proposed a clustered voltage scaling technique which assumes two different voltages available and finds the optimal voltage assignment to each cell considering the

overhead of level shifting cells [Usami95]. Johnson et al. proposed a multiple voltage scheduling technique for reducing the energy consumption of a data path circuit considering an energy overhead of level shifting circuits [Johnson97].

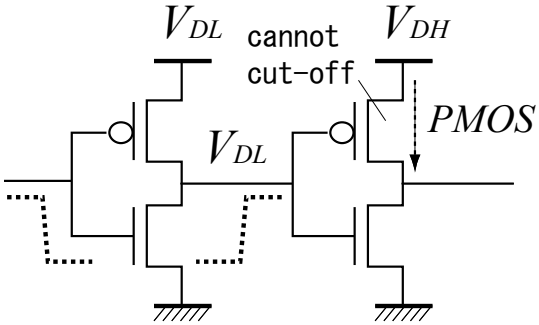


Figure 6-3. Static Current in Low Voltage Circuits

3.2 Power-Delay Tradeoff

Although lowering the supply voltage is the most effective way for reducing the energy consumption of SoC chip, this causes an increase of circuit delay, τ, which determines the maximum clock frequency of synchronous circuits. The delay τ of a CMOS circuit can be approximately formulated as (3.1),

$$\tau \propto \frac{V_{DD}}{(V_{DD} - V_{th})^2} \cong \frac{1}{V_{DD}} \tag{3.1}$$

where V_{th} is the threshold voltage of CMOS transistors used in the circuit.

Basically, we have the following three ways for lowering the operating voltage without sacrificing the performance of the system.

1. Parallelize tasks so that the performance does not degrade even in a low voltage operation. We refer this approach as *static voltage scaling*.
2. Use the maximum available supply voltage for gates on a critical-path and use a lower supply voltage for the other gates. We refer this approach as *multiple voltage assignment*.
3. Lower the clock frequency and operating voltage when the maximum performance is not needed. We refer this approach as *dynamic voltage scaling*

3.2.1 Static Voltage Scaling

Suppose we have four sequential tasks as shown in Figure 6-4 (a) and we have two processing units each of which can complete each task per a unit time T_{UNIT} when 5.0V is used. If the tasks can be concurrently run on the processing units as shown in Figure6-4 (b), the clock frequency and operating voltage of the processing units can be reduced by half without degradation of system performance. Although switching activities per a unit time may increase up to twice, we can reduce the number of cycles and V_{DD} by half. As a result, energy consumption can be quarter without performance degradation.

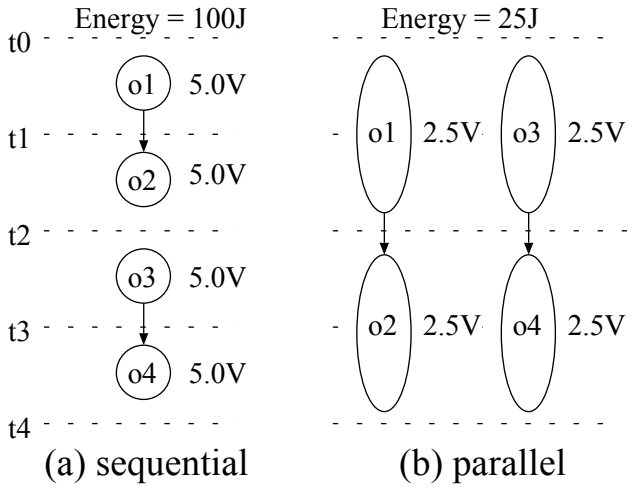


Figure 6-4. Energy Reduction by Parallel Computation

A lot of researchers have proposed methods that incorporate architectural-level voltage scaling. Chandrakasan et al. proposed HYPER-LP which optimizes dataflow graph generated from a target application program for reducing the power consumption of data-path circuits [Chandrakasan95]. Other methods try to transform the target circuit during scheduling, module selection, resource binding, etc., for minimizing power consumption [Raghunathan94][Raghunathan95][Coodby94][Kumar95][Martin95]. All of the methods mentioned above try to exploit parallelism in the algorithm to shorten critical paths so that lower supply voltage can be used. Although this is a very attractive approach, parallelization of the computation is generally difficult because some computations are inherently sequential.

3.2.2 Multiple Voltage Assignment

Most voltage scaling techniques assume that the circuit operates at a single supply voltage. Although substantial energy savings can be achieved with a single minimum supply voltage, one cannot always take full advantage of available schedule slack to reduce the supply voltage. Since path delays in the circuit are not uniform, supply voltage of gates on a non-critical path can be lowered until the path delay meets with the clock period. When there are nun-uniform path delays, the critical path delay determines the clock period. In this case, non-critical paths use only part of a clock period. The slack time within these clock periods goes to waste. Additional voltages make it possible to use the entire clock period. The basic idea is to assign lower V_{DD}s to the non-critical paths in a way that the delays of the paths meet with the clock period as shown in Figure6-5.

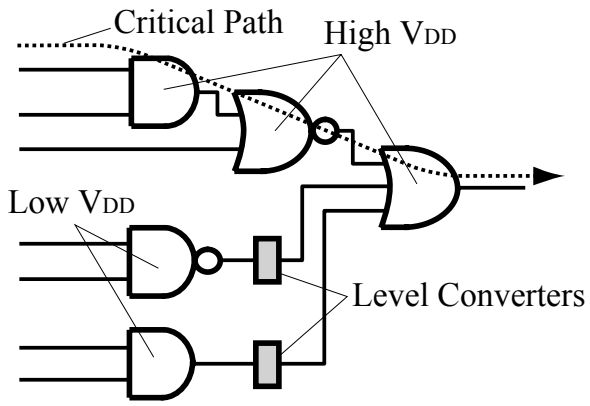
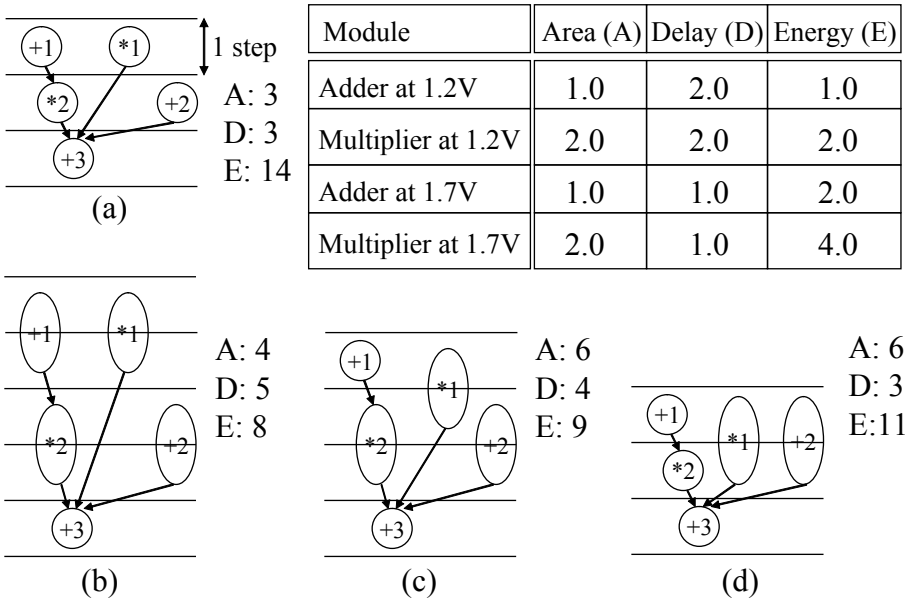


Figure 6-5. An Example of the Multiple Voltage Assignment

Usami et al. proposed a voltage assignment algorithm which finds the optimal voltage assignment to each cell considering a level shifting cell between different voltages [Usami95]. The algorithm performs backward graph-traversal for a given netlist from the primary outputs toward the primary inputs using the Depth-First-Search (DFS) algorithm. Each time the algorithm visits a cell and tries to replace a high V_{DD} cell with low V_{DD} cell. If the timing constraint is still met even after the replacement, the cell is replaced. This process is repeated until all the cells are visited. Their experiments demonstrated that the energy consumption can be reduced by 20% using two voltages 5V and 3V.

The idea can be extended to a multiple voltage datapath scheduling technique in high level synthesis. The main idea is to minimize energy

consumption by assigning operations to time steps with various supply voltages under a given time or resource, or both constraints. We use Figure 6-6 to illustrate the multiple-voltage scheduling technique. Assume the energy consumption of an addition operation is 1.0 at 1.2V and 2.0 at 1.7V. It requires 2 time steps at 1.2V but only 1 step is sufficient when 1.7V is applied. The area required for the adder module is 1.0. Similarly, the energy consumption of a multiplication operation is 2.0 at 1.2V and 4.0 at 1.7V. It requires 2 time steps and 1 step at 1.2V and 1.7V, respectively. The area required is 2.0. Suppose we have a control flow graph as shown in Figure 6-6 (a). It needs 3 steps and the energy consumption is 14. Since we can share the resources, we only need one adder circuit and one multiplier circuit in this case. As a result, the area required is 3. Since operations *1 and +2 are not located on a critical path, we can assign lower voltage to them as shown in Figure 6-6 (d). In this case, energy consumption can be reduced to 11. If we relax the time constraint to 5, we can reduce the energy consumption to 8 as shown in Figure 6-6 (b). This idea is extended in the following papers [Johnson97][Raje95][Lin97][Chang96] so as to fit with more practical situations.



Module	Area (A)	Delay (D)	Energy (E)
Adder at 1.2V	1.0	2.0	1.0
Multiplier at 1.2V	2.0	2.0	2.0
Adder at 1.7V	1.0	1.0	2.0
Multiplier at 1.7V	2.0	1.0	4.0

Figure 6-6. Multiple-Voltage Scheduling in High Level Synthesis

Raje et al. proposed a datapath scheduling technique which schedules the datapath operations, selects voltages from a predetermined set of voltages and assigns the voltages to the datapath operations simultaneously so as to

minimize power consumption [Raje95]. Lin et al. used and integer linear programming approach to schedule datapath operations, choose voltages from a list of candidates, and assign voltages to each operations considering timing and resource constraints together [Lin97]. Johnson et al. used an integer linear programming approach to choose voltages from a list of candidates, schedule datapath operations, and assign voltages to each operations considering the energy overhead of level converters [Johnson97]. Chang et al. proposed a dynamic programming approach to optimize non-pipelined datapaths and modified list scheduler to handle functionally pipelined datapaths [Chang96].

3.2.3 Dynamic Voltage Scaling

More aggressive approach is dynamic voltage scaling. Since the computational load is not constant during the execution of given tasks, we can control computational power according to the computational load. The basic idea is assigning different operating voltages to the tasks in a way that any of the tasks does not violate a timing constraint. The assignment can be done statically or dynamically.

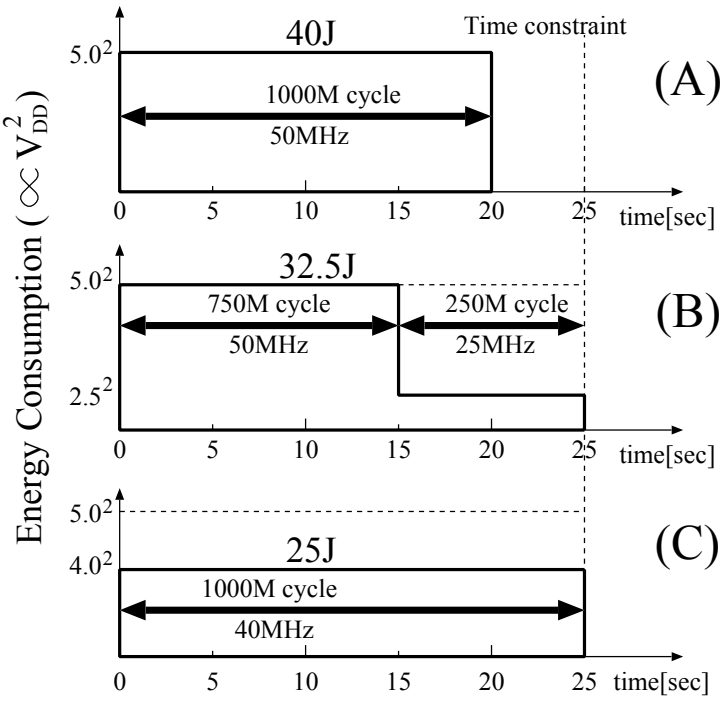


Figure 6-7. Motivational Example

Figure 6-7 shows motivational example of the dynamic voltage scaling. Suppose we have a processor which uses three different supply voltages, 5.0V, 4.0V, and 2.5V. A task which takes 1 billion cycles to complete runs on the processor. The energy consumptions for the task are 10nJ/cycle, 25nJ/cycle and 40nJ/cycle at 2.5V, 4.0V and 5.0V, respectively. The computational speeds of the processor at 5.0V, 4.0V, and 2.5V are 50 million cycles per second, 40 million cycles per second, and 25 million cycles per second, respectively. This assumption follows the Equations (2.1), (2.2) and (3.1).

In Figure 6-7 (A), the processor uses the maximum supply voltage, 5.0V, for the entire execution of the task. In this case, the total energy consumption is 40J. If the processor uses 2.5V and 5.0V in a way that the completion time of the task meets with a given time constraint, the energy consumption can be reduced to 32.5J as shown in Figure 6-7 (B). Figure 6-7 (C) shows the best case of this example. If the processor uses a single supply voltage which adjusts the completion time just to the time constraint, the total energy consumption is minimized.

In [Ishihara98], Ishihara and Yasuura proved the following theorem; if the processor uses a voltage, v_{ideal}, and completes a given task just at a timing constraint T_{const}, the v_{ideal} is the ideal voltage which minimizes energy consumption for the task. The example shown in Figure 6-7 demonstrates that reducing the energy consumption of the processor is fundamentally equivalent to exploiting idle intervals of the processor. Thus, we should first identify sources of idle intervals to efficiently reduce the power consumption of the processor. There are three major sources as follows;

1. The first one occurs when a system is not tightly designed for a given processor. In other words, there is a room for design change or improvement such as introducing more tasks, replacing certain tasks with their version ups, using lower performance processors and so on.
2. The second source comes from a nature of a fixed-priority scheduling. The idle intervals inhere in the fixed-priority scheduling, because the priorities statically assigned to the tasks are not always optimal for the tasks.
3. The third source comes from run-time variation of execution time. Since most of tasks complete its execution much earlier than the worst case execution time, the slack time will be yielded depending on input data for the task.

Consider the three tasks given in Table in Figure 6-8. T_i, D_i and C_i denote period, deadline and the worst case execution time (WCET) of each task, respectively. Priorities are assigned in row order as shown in the fifth

column of the table. Assume all tasks are released simultaneously at time 0. A typical schedule, which assumes that tasks run at their WCETs (C_i), is shown in Figure 6-8 (a). Note that this system is designed to meet its schedulability. For example, if τ2 takes a little longer to complete, τ3 would miss its deadline at time 100. Even though the system is tightly designed, there are still some idle time intervals, as shown in Figure 6-8 (a). At time 160 in the figure, when the request for τ2 arrives, the run-time task scheduler knows that there will be no requests for any tasks until time 200, which is the time when requests for τ2 and τ3 will arrive. As a consequence, we can save power by reducing the speed of the processor by lowering the clock frequency and supply voltage. When tasks are completed earlier than their WCET, we have more chances to apply the same mechanism. For the example of Figure 6-8 (b), we can slow down the processor at time 50 because the first instances of τ2 and τ3 complete their execution earlier than the second request for τ1 arrives. Since the execution time of each task frequently deviates from its WCET during the operation of the system, we have many chances to slow down the processor as shown in Figure 6-8.

	T_i	D_i	C_i	*Priority*
τ1	50	50	10	1
τ2	80	80	20	2
τ3	100	100	40	3

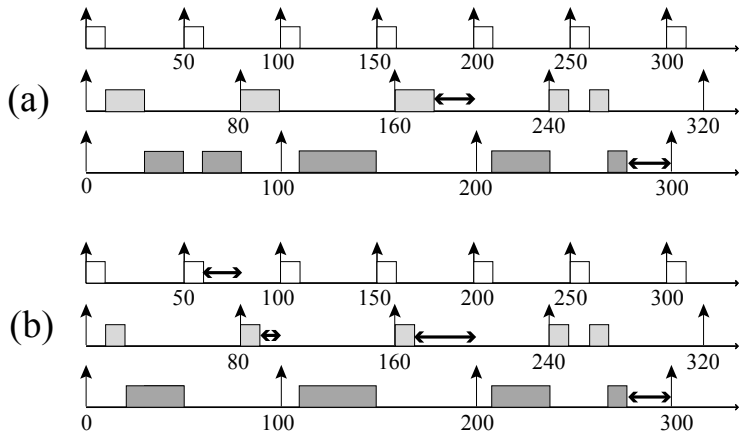


Figure 6-8. An Example of Task Scheduling on a Variable Voltage Processor

Weiser et al. proposed a scheduling method for dynamically variable voltage processors [Weiser94]. Yao et al. proposed real-time task scheduling

methods for the dynamically variable voltage processors [Yao95]. Both of them assume a fixed amount of execution time and exploit the first source of idle intervals only.

In [Shin99], Shin et al. proposed a fixed-priority scheduling method which exploits the second and third sources of idle intervals mentioned above. They extended this work and proposed off-line and on-line algorithms for exploiting all of idle intervals mentioned above [Shin00]. The off-line algorithm finds the lowest possible voltage which guarantees time constraints of all tasks. The on-line algorithm dynamically varies the processor speed along with the supply voltage in order to exploit execution time variations and idle intervals.

In [Okuma99], Okuma et al. proposed a real-time task scheduling algorithms for the dynamically variable voltage processor. Their approach based on the Earliest Deadline First (EDF) algorithm. Similar to [Shin00], their approach exploits the first and third sources of the idle intervals mentioned above. However they assume to choose voltages from a limited number of candidates, while [Shin00] assumes to use continuous values of voltage and clock frequency which is practically impossible.

3.3 Power-Reliability Tradeoff

Since the voltage scaling technique reduces voltage margins, it is impossible to discuss about low-power design techniques without considering reliability issues. Most circuit designers have to determine supply voltage of the target circuit to ensure that all circuits operate correctly even in the worst-case operating environment. There are three measure voltage margins as follows [Austin04].

1. Process Margin
   This ensures that performance uncertainties resulting from manufacturing variations in transistor do not prevent slower devices from completing computation within a clock period.
2. Ambient Margin
   This ensures correct operation at the worst-case temperature.
3. Noise Margin
   This protects against a variety of noise sources that introduce uncertainty in supply and signal voltage levels, such as di/dt noise in the supply voltage and cross-coupling noise in logic signals.

The sum of these voltages defines the minimum supply voltage that ensures correct circuit operation even in the worst-case condition. As mentioned before, the energy consumption of CMOS circuit is quadratically

proportional to the supply voltage. Therefore, it is clear that there is a trade-off between reliability and energy consumption.

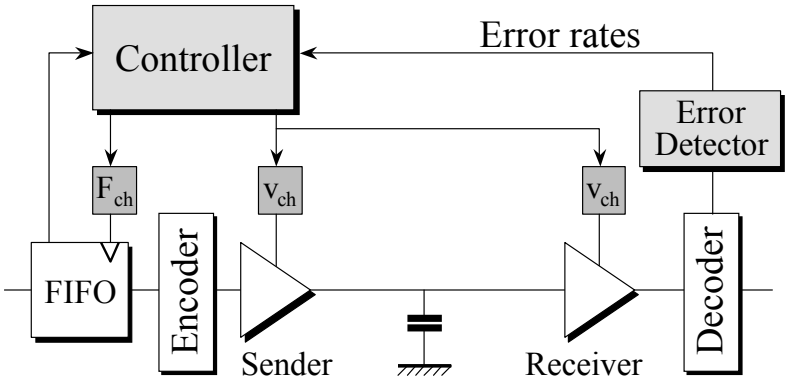


Figure 6-9. Dynamic Voltage Scaling for Reliable Data Transmission

Worm et al. proposed an interconnect system which uses low-swing signaling, error detection codes, and a retransmission scheme [Worm02]. This technique optimally finds the interconnect voltage swing and frequency with subject to workload requirements and signal to noise conditions. The most straightforward way to reduce the energy consumption for the communication is lowering the voltage swing of signals propagated through interconnects. This however causes an increase of sensitivity to noise sources because of the decreased noise margins. Their technique monitors bit error rates of the interconnect on the fly as shown in Figure 6-9 and dynamically finds the optimal swing level which minimizes energy consumption while satisfying the reliability constraint. Their simulation results show that the energy consumption can be reduced by 56% over a conventional interconnect with more robustness to large variations in actual workload, noise and technology quality.

Bertozzi et al. evaluated energy efficiency of several error resilient techniques such as error correcting codes, a data retransmission technique and so on [Bertozzi02]. Their experiments demonstrated that retransmission strategies are more effective than the error-correction-based technique in terms of energy efficiency.

Austin et al. proposed Razor, a voltage scaling technique based on dynamic detection and correction of circuit timing errors [Austin04]. The technique eliminates unnecessary voltage margins that the traditional worst-case design methodologies require. In some cases, computations may fail and require additional time and energy for recovery. However, the overall

computation consumes significantly less energy than traditional worst-case design.

3.4 Commercial Products

There has been a lot of power management software released before. Early power management software used the BIOS to determine whether a device had been idle long enough to shift a sleep state. With the introduction of Advanced Power Management (APM) the OS began to control the power settings and timings. With the Advanced Configuration and Power Interface (ACPI) specification, all power management moved from the BIOS to the hardware and operating system. In today's low-power oriented computer systems, chipsets support ACPI power and thermal management functions to control various system-level and processor-level power and sleep states, and they also still support APM. However, neither APM nor ACPI supports dynamic voltage scaling of chipsets. Recently, many computer systems including laptop PCs, PDAs, cellular phones, and etc. introduced the dynamic voltage scaling technique. The following power management software support dynamic voltage scaling.

- SpeedStep™, Extended SpeedStep™ (Intel)
- PowerNow! ™ (AMD)
- LongHaul™ (VIA Technologies)
- LongRun™, LongRun2™ (Transmeta)
- SmartReflex™ (TI)
- IEM™ (ARM)

Most of the above software products are based on the dynamic voltage scaling techniques mentioned in this section. Some of them also support a dynamic body biasing technique which can dynamically control the threshold voltage of transistors for reducing the leakage power consumption of a chip. The detailed explanation of the dynamic body biasing technique will be provided in Section 6.

3.5 Conclusions

In this section we addressed several techniques for lowering supply voltage of chips considering voltage compatibility, a power-delay tradeoff and a power-reliability tradeoff. As mentioned above, lowering supply voltage has the biggest impact on power reduction. The techniques can be applied to many kinds of SoC implementations like multi-chip module (MCM), network on chip (NoC), system in package (SiP), chip multi

processor (CMP) and so on. However, it becomes more difficult in future to control supply voltage due to the reliability issues. Breakthrough will be appeared if we can tolerate negative effects of process variations, temperature variations, soft errors and noises even in ultra low-voltage operation.

4. TECHNIQUES FOR REDUCING SWITCHING ACTIVITY

Lowering the switching activity is a very promising way of decreasing the power consumption. There are numerous researches on this issue. In this section, we introduce system level approaches for reducing the switching activity. System level switching activity reduction can be categorized as follows:

- Turn off unused HW modules.
- Adjust datapath, the bit width of buses and operational units in a system.
- Trade precision for low power (Use narrow bit width).
- Compiler based instruction scheduling

Practical strategies we pick up in this section are shutting down unused modules, adjusting datapath width to minimize power consumption and compiler optimization techniques for reducing the switching activity.

There are two main shutting down strategies: *clock gating* and *power gating* as summarized in Figure 6-10. Power gating is mainly used for reducing leakage power. Section 6 describes the power gating in more detail. The best-known technique for reducing the switching activity is clock gating. Clock network power can account for as much as 75 percent of the total switching power of a chip, and sequential cells driven by clocks can account for as much as 70 percent of the total clock power. Clock gating essentially disables the clock to a circuit to save power by both preventing unnecessary activity in logic modules and by eliminating power dissipation on clock network. Using a simple AND or OR gate (depending on the edge on which flip-flops are triggered) with the enable and clock signals as inputs produces a gated clock as output. One can also employ a level-sensitive latch to hold the enable signal from the active edge until the inactive edge of the clock. Clock gating can be applied in either fine-grained or coarse-grained manner. Fine-grained allows us to reach miscellaneous small units in clock sinks and aggressively save their dynamic power even for a few cycles. Coarse-grained gating saves power from higher level of the clock tree by removing all clock switching from its down-stream units.

	Clock Gating	Power Gating
Hardware support	Easy	Difficult
Overhead (power & area)	Small	Large
Overhead (delay)	Small	Large
Energy efficiency	Moderate	Large

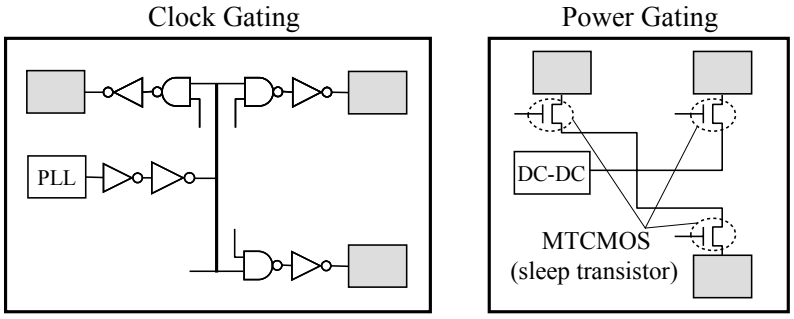


Figure 6-10. Comparison of Clock Gating and Power Gating

Another strategy for reducing switching activity is datapath width adjustment. Since datapath width, the bit width of buses and operational units in a system, strongly affects the size of circuits and memories in a system, the power consumption of a system also depends on the width of the datapath. In design of embedded systems and System-On-a-Chip (SOC), designers have to consider the trade off among system performance, cost and power consumption. Bitwidth of data, the length of data, computed in the system is one of the most important design parameters related with performance, cost and power of the system. The bitwidth of datapath and the size of memories strongly depend on the bitwidth of data. Providing more datapath width for computation than required, will consume more dynamic power and leakage power than necessary by the extra bits.

Typical algorithms defined in C/C++ or SystemC will initially not contain definitions of the actual bit width for operations and storage elements. For algorithm selection, the design team often relies on floating point and straight integer calculations. Based on the stimulus which is applied to the design under optimization users can assess the minimum and maximum values on specific operations and then choose the optimal bit width accordingly. This allows users to understand the impact of bit width on energy and is a step towards trade offs between *quality*, which may be higher in a video application using higher bit width, vs. energy which decreases with lower bit width in the operations. In quality driven design, both higher and lower bits of data can be reduced. From the requirements on the output

quality, lower bits of data may be omitted in the datapath width adjustment (See Figure 6-11). This means that there is potential for further energy reduction by decreasing computation accuracy.

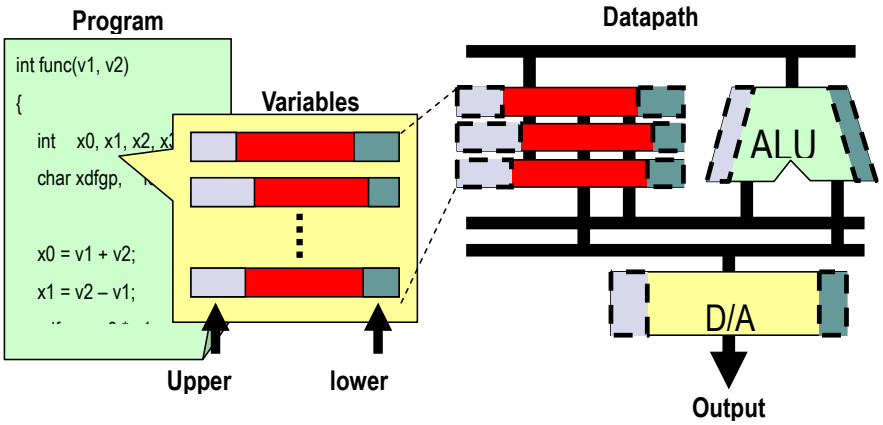


Figure 6-11. Datapath Adjustment

In this section, we describe dynamic power management by using the shutting down strategy, the datapath width adjustment strategy, and instruction scheduling.

4.1 Dynamic power management (DPM)

System level dynamic power management (DPM) has gained considerable attention in recent years as a way to save energy in devices that can be turned on and off. DPM dynamically reconfigures systems to provide the requested services and performance levels with a minimum number of active components or a minimum load on such components. The fundamental premise for the applicability of DPM is that systems and their components experience non-uniform workload during operation time and that it is possible predict, with a certain degree of confidence, the fluctuations of workload. There are two power reduction methodologies when idle modes: voltage scaling with frequency scaling and clock gating. Only clock gating methodology is introduced.

The control procedure is often called policy. An example of a simple policy, ubiquitously used for laptops and palmtops, is the timeout policy, which shuts down components after a fixed inactivity time, under the assumption that it is highly likely that a component remains idle if it has been idle for the timeout time. Power could be shut off or gated to functional

blocks when operating in a standby mode and restored as needed. The gated circuit would not dissipate any power when turned off. Additional circuit would be required to monitor the need for these functional blocks. A problem with power gating is the latency between when the signal to turn a unit on arrives and when the unit is ready to operate. Retention flip-flops on an isolated power supply could be used to save the logic state of all sequential elements when a chip is powered down, eliminating the need to reinitialize the device when it comes out of standby mode. Some products support multiple levels of standby (soft off, nap and sleep) which differ in terms of the amount of power saving and latency (See Figure 6-12).

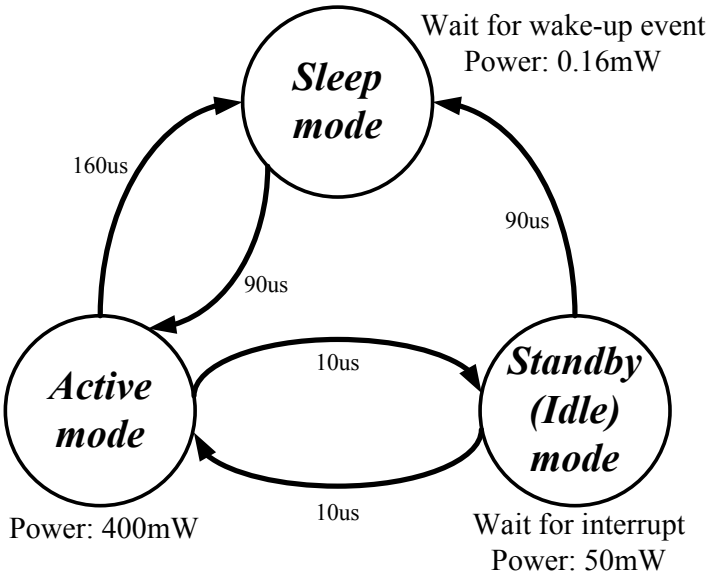


Figure 6-12. Dynamic Power Managament

4.2 Datapath width adjustment (Bit-width optimization)

Processor-based systems treat various data with different bit width. It is efficient in power reduction not only to determine datapath width statically but also to control the active datapath width dynamically.

First, we introduce static optimization, which adjusts datapath width. Bit-width analysis is performed to extract information on the required bit width of variables in programs and algorithms. For hardware design, using the result of bit-width analysis, one can determine the length of registers, the

size of operation units, and the width of memory words on the datapath of a system to minimize the meaningless power consumption by the useless bits. Shorter registers and operation units reduce switching activity and the leakage of extra bits on the datapath. However, the trade-off between power consumption and execution time needs to be resolved. Generally, narrowing the datapath width reduces the area and power of the processor, but degrades the performance. The number of execution cycles increases, since some single-precision operations should be replaced with double or more precision operations in order to preserve the accuracy of the computation. Single-precision operations are those whose precision is smaller than that of the datapath width. For example, an addition of two 32-bit data is a single-precision operations whose datapath width is equal to or greater than 32 bits, while it is a double precision operation on 16-bit processors. Changing the datapath width affects the size of data memory (RAM) and instruction memory, which is mostly implemented by ROM in embedded systems. Let us consider a program including two variables *x* and *y*, and assume that two variables *x* and *y* require at most 18 bits and 26 bits, respectively (see Figure 6-13). When the datapath width is 32 bits, two words are required to store these two variables, and the amount of the data memory is 64 bits. Since the minimum bit size required to store the variables is only 44 bits (18+26), 20 bits of the memory (about 30%) are unused. By reducing the datapath width to 26 bits, one can reduce the unused bits to 8 bits. Unused bits, however, increase to 31 bits, if a 25-bit datapath is adopted, because *y* requires two words. When the datapath width is 9 bits, two words and three words are required for *x* and *y*, respectively, and the unused area is only 1 bit. Many unused bits in the data memory can be eliminated by datapath-width optimization.

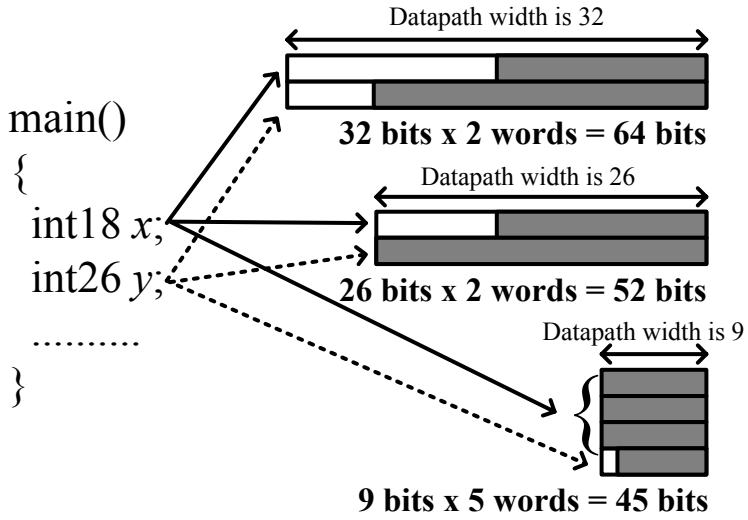


Figure 6-13. An Example of Datapath Width Adjustment

Second, dynamic approach, which control active datapath width, is introduced. This approach is called *value-based clock gating*. There is a fact that "narrow-width" data is common not only in multimedia codes, but also in more general workloads. For example, over half the integer operation executions require 16 bits or less on a 64-bit processor. Basic mechanism to reduce power consumption is operand-value-based clock gating to turn off portions of memories, buses, and arithmetic units that will be unused by narrow-width operations. This optimization results in around 50% reductions in the data bus and integer unit power consumption. By Appling this for data memory, 80% power reduction can be achieved. However, this approach requires hardware cost for detecting dynamically operation widths and turning off the unused units. As shown in Figure 6-14, if there is a 7-bit width data, only the lower data memory (D0) is accessed.

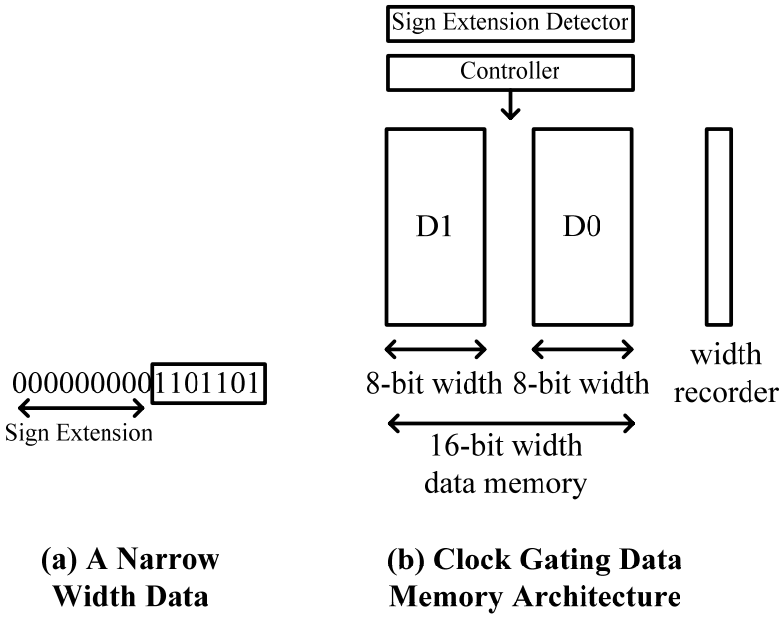


Figure 6-14. A Data Memory Example Using Operand Based Clock Gating

4.3 Compiler Optimization

Compiler optimization is also effective for reducing the switching. In [Tomiyama1998], they proposed an instruction scheduling technique to reduce power consumption due to off-chip driving. Their technique reduces transitions on a data bus between an on-chip cache and a main memory, and as a result, power consumed by off-chip drives in the main memory is reduced. Let us consider an example in Figure 6-15, and assume 8-bit instruction width and 32-bit cache line size. There are four instructions (a)-(d) in the memory block. When the memory block is sent to the cache, the instruction (a) is sent first. At the time, four bits switch from high- to low-level. At the next cycle, (b) is sent to the cache and six bits switch to opposite level. As a result, the cache miss invokes twenty four transitions totally in the data bus. If changing the positions of two instructions (b) and (c) keeps the meaning of the program, it reduces bus transitions by 25%, from twenty four to eighteen bus transitions (See Figure 6-15). Thus the instruction scheduling can reduce transitions on the bus. Tomiyama et al. reported that the scheduling algorithm achieves significant reduction in transitions on the data bus, up to 28% of reduction, and runs efficiently.

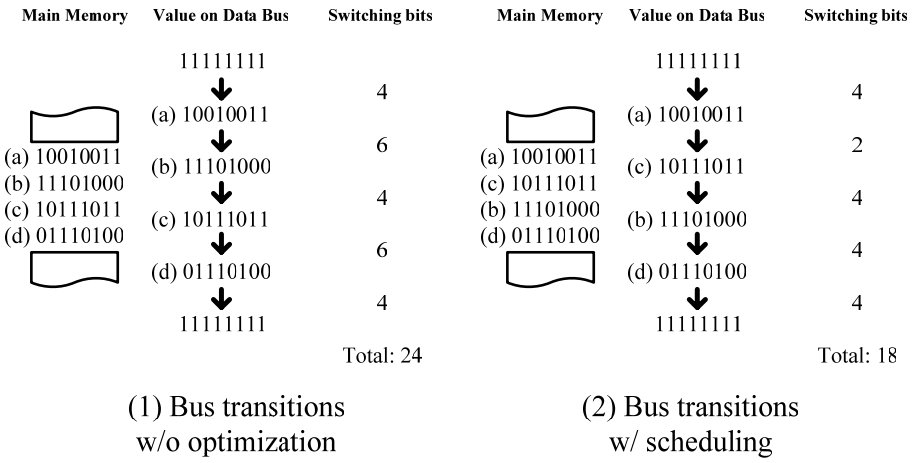


Figure 6-15. An Example of Instruction Scheduling for Low Power

4.4 Commercial Products

The Pentium 4 processor uses the clock gating technology. Every unit on the chip has a power reduction plan, and almost every functional unit block contains clock gating logic.

4.5 Conclusions

In this section, we summarized system level switching activity reduction strategies. The basic strategies are clock gating and datapath width adjustment. Analyzing statically and dynamically system requirements, unnecessary switching activity reduction can be achieved.

5. TECHNIQUES FOR REDUCING THE PRODUCT OF SWITCHING ACTIVITY AND A LOAD CAPACITANCE

A major contributor to the system budget is the memory-processor interface. Ko et al. mentioned that the power dissipation of an external memory access is at least an order of magnitude higher than that of an on-chip access [Liu94][Ko98]. For this reason, a lot of techniques for reducing

energy consumption of the off-chip buses have been proposed. The basic idea is reducing the switching activities (*SA*) of hardware modules whose load capacitance (*CL*) is large even if the *SA*s of low-*CL* modules are increased. Suppose we have a processor system including a CPU core, cache memories, an off-chip memory, and a processor-memory interface as shown in Figure 6-16. The energy dissipation of the memory-processor interface, $E_{interface}$, can be expressed by (5.1),

$$E_{interface} = N \cdot \left(E_{address} + E_{data} + E_{memory} + E_{overhead} \right) \tag{5.1}$$

where N, $E_{address}$, E_{data}, E_{memory}, and $E_{overhead}$, represent the number of memory accesses, the energy dissipation in address buses per access, that in data buses per access, that in a memory module per access and energy overhead per access, respectively. There may exist the energy overhead if the memory-processor interface is modified for reducing the energy consumption in off-chip buses. As one can see, we can reduce the energy dissipation of the processor-memory interface by decreasing N, $E_{address}$, E_{data}, E_{memory}, and $E_{overhead}$. The problem of minimizing the total energy consumption of the processor system is basically equivalent to finding the best tradeoff point between on-chip computational energy and off-chip communication energy.

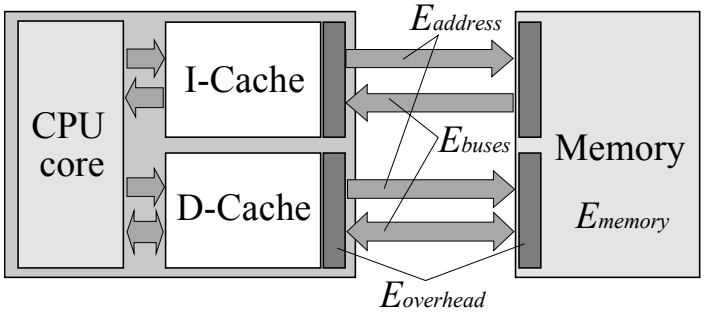


Figure 6-26. Energy Dissipation of Processor-Memory Interface

There are the following three major approaches for reducing the energy required for the communication between a memory and a processor.

- Cache miss reduction
- Bus encoding
- Code compression

5.1 Cache Miss Reduction

Since cache miss rate is associated with the number of off-chip memory access, reducing cache miss rate leads to a reduction of the energy dissipation for the off-chip memory accesses. The most straightforward way for reducing the cache miss rate is to employ larger cache memory on a chip. Many techniques have been proposed for optimizing cache configuration considering tradeoff between energy consumption of off-chip memory and cache memory [Su95][Hicks97][Li98][Shine99][Malik00]. All these techniques are based on the fact that while a bigger cache consumes more energy per access, it can reduce the number of cache misses and as a result can reduce the energy consumption for the off-chip accesses. Suppose we have a processor with on-chip cache memory which can be resized for the target application as shown in Figure 6-17.

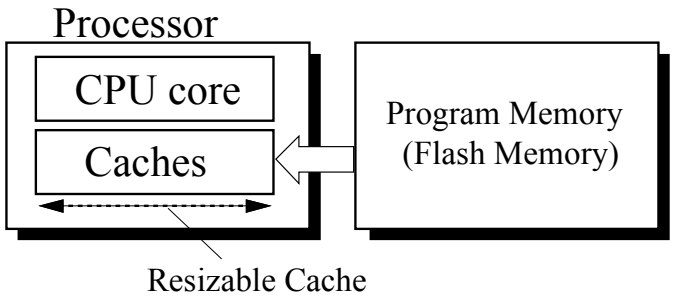


Figure 6-17. An Example of Resizable Cache

If we optimize the cache size for the target application, the energy consumption for memory accesses can be drastically reduced. For example, based on the experiment in [Ishihara05], the optimal cache size for the SPEC95 benchmark program, “Compress”, is 2kB as shown in Figure 6-18. If we use the 4kB cache instead of 2kB power consumption of the cache becomes very large. Conversely, if the 1kB cache is used, the power consumption of off-chip memory becomes huge due to the large number of cache misses. In the optimal case, the power consumption can be reduced by 85% compared to the result for 1kB cache memory. Note that the leakage power of the cache memory is assumed to be 10% of its dynamic power consumption.

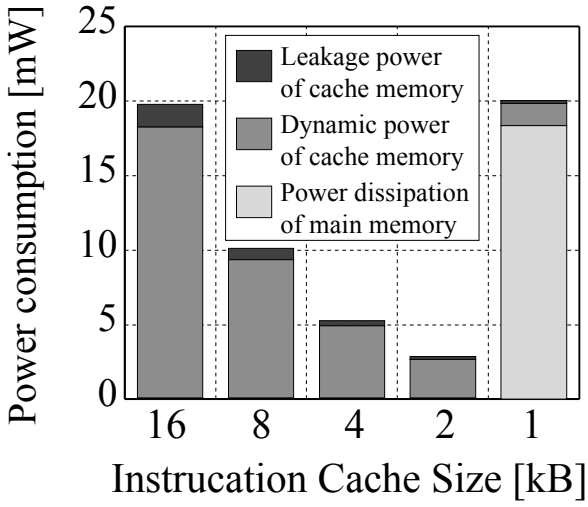


Figure 6-18. Cache Optimization for Low Power

Li and Henkel proposed *Avalanche* framework which simultaneously evaluates the tradeoffs of energy dissipations of caches and main memory [Li98]. The trade-off between system performance and energy dissipation is also explored in the framework. Their experiments demonstrated significant improvements (up to 95% energy saving) in energy dissipation.

Another approach to reducing the number of cache misses is a compiler-based approach [McFarling89][Hwu89][Tomiyama96][Panda96] [Hashemi97][Ghosh99]. The idea is to modify the place of basic blocks, procedures, or global variables in the address space so that the number of cache conflict misses is minimized. This can significantly reduce the number of cache misses and energy consumption of memory subsystems. We first explain the idea behind the typical code and data placement technique. Consider a direct-mapped cache of size C (= 2^m words) whose cache line size is L words, i.e., L consecutive words are fetched from the main memory on a cache read miss. In a direct-mapped cache, the cache line containing a word located at memory address M can be calculated by ($\lfloor M/L \rfloor$ mod C/L). Therefore, two memory locations M_i and M_j will be mapped onto the same cache line if the following condition holds,

$$\left(\left\lfloor \frac{M_i}{L} \right\rfloor - \left\lfloor \frac{M_j}{L} \right\rfloor \right) \bmod \frac{C}{L} = 0 \qquad (5.1)$$

Several code and data placement techniques have used the above formula [5.6-5.13]. Assume a direct mapped instruction cache with 4 cache-lines,

where each cache-line is 32 bytes as shown in Figure 6-19. Functions A, B, C and D are placed in the main memory as shown in the left side of Figure 6-19. If functions A, B, and D are accessed in a loop, conflict misses occur because A and D are mapped onto the same cache line. If the locations of C and D are swapped as shown in the right side of Figure 6-19, the cache conflict is resolved. Code placement techniques modify the placement of basic blocks or functions in the address space so that the total number of cache conflict misses is minimized. Similar to the code placement techniques, data placement techniques modify the placement of global variables in the address space so as to reduce the number of data cache misses.

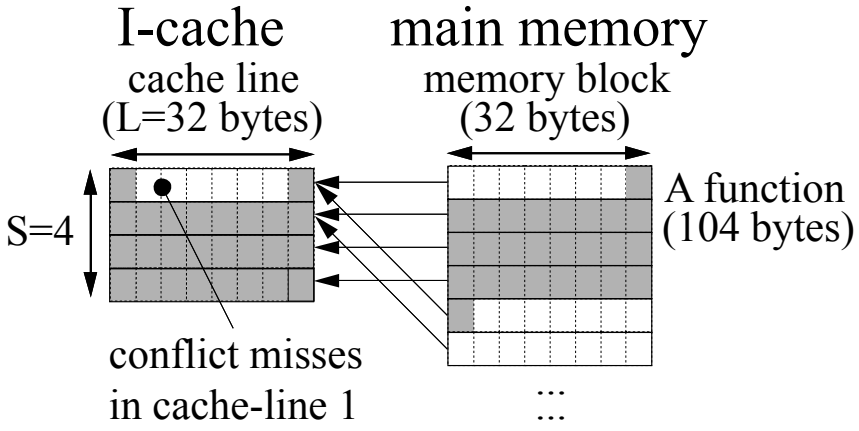


Figure 6-19. An Example of Code Placement

Kulkarni et al. proposed a data placement algorithm which finds the optimal locations of global variables in the main memory [Kulkarni01]. The algorithm also explores different cache sizes considering trade-offs among performance, energy consumption and chip area. In the first step, they measure the cache miss rates for different cache sizes. Once the miss rates are obtained, the algorithm performs data placement for each cache size and estimate the energy consumption including energies for on-chip accesses and off-chip accesses. Depending on the design constraints, the designer can either choose a lower power solution with some overhead in size and vice versa. Their experiments demonstrated that the total energy consumption can be reduced by 10.6% with 26% performance overhead and 7% area overhead.

Scratchpad memory can be used as a design alternative for the on-chip cache memory. Current embedded processors particularly in the area of multimedia applications and graphic controllers have on-chip scratchpad memories. In cache memory systems, the mapping of program elements is

done during runtime, while in scratchpad memory systems this is done by the programmer or the compiler. Unlike the cache memory, the scratchpad memory does not need tag search operations and, as a result, it is more power efficient than the cache memory if programmers or compilers can optimally allocate code and data on the scratchpad memory.

Ishihara and Yasuura proposed a code allocation technique which finds a size of an on-chip scratchpad memory and a code allocation to the scratchpad memory simultaneously so as to minimize the total energy required for fetching instructions [Ishihara00]. Their experiments showed that the energy consumption for the instruction fetching can be reduced by 50%. Benini et al. presented a novel solution for the design hierarchy of low-power embedded systems [Benini00]. The idea is mapping the most frequently accessed data onto a small memory, called application-specific memory (ASM) which is placed vary close to the processor. The experimental results on a set of typical embedded programs have shown that the energy consumption can be reduced by 68% with respect to equivalent caches having different sizes, organizations and configurations. Banakar et al. proposed an approach for selection of on-chip memory configuration from various sizes of cache and scratch pad memories [Bankar02]. Their experiments show that scratchpad based compile-time memory outperforms cache-based run-time memory on almost all aspects. For example, the total energy consumption of scratchpad based systems is less than that of cache-based systems by 40% on an average.

5.2 Bus Encoding

Bus encoding techniques reduce communication power by changing the format of the information in a way that the total communication power is minimized. The basic strategy is to reduce switching activity of off-chip buses by encoding data transmitted between a processor and a memory. We have to consider a tradeoff between the energy consumed in buses and the energy overhead of encoding and decoding circuits. Suppose we have an original data format, *Format-A*, and low-switching format, *Format-B* as shown in Figure 6-20. Energy consumption for sending data using *Format-A* and *Format-B* is *EA* and *EB*, respectively. The energy overhead for encoding and decoding (i.e., translating *Format-A* into *Format-B* and vice versa) is $E_{overhead}$. Bus encoding techniques are effective only when the following inequality holds,

$$EA > EB + E_{overhead} .$$

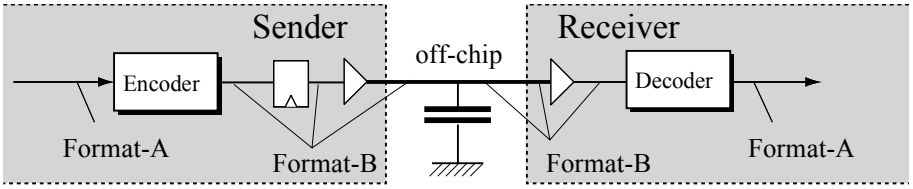


Figure 6-30. Low-Power Bus Encoding

The bus-invert coding is one of the most popular approaches [Stan95]. In the bus-invert coding, if the Hamming distance (the number of switched bits) between the new pattern to be transferred and the old one currently on the bus is larger than half the bus width, the new pattern is transferred with each bit inverted. An additional invert bit is used to inform the receiver side whether the pattern is inverted or not. The experiments demonstrated that the bus-invert coding technique decreases the I/O peak power dissipation by 50% and the I/O average power dissipation by 25%.

For instruction address patterns, where consecutive patterns are often sequential, the Cray code is efficient [Su94]. The Gray code has only one-bit difference in consecutive number for addressing. Due to locality of program reference, Gray code addressing can significantly reduce the number of bit switches. The experimental results showed that for typical programs running on a RISC microprocessor, using Gray code addressing reduce the switching activity at the address lines by 30-50% compared to conventional binary code addressing.

In the *T0 code* [Benini97], the bus transitions are further reduced by freezing the address lines when consecutive patterns are detected to be sequential. An extra bus line is employed to inform the receiver side whether or not the current pattern is sequential.

In special purpose applications, where the information about the sequence of patterns is available a priori, the characteristics of patterns can be exploited to efficiently reduce bus transitions. The Beach Solution [Benini97-2] makes clusters of bus lines based on statistical information of address patterns and then generates an encoding function for each cluster such that the encoded version of each cluster results in less transitions.

For data address patterns which are less sequential than instruction address patterns and less random than data patterns, the Partial Bus-Invert code [Shin98] performs better. It applies the bus-invert coding to a pre-defined sub-group of bus lines thereby avoiding unnecessary inversion of relatively inactive and/or uncorrelated bus lines. The experiments on benchmark examples indicate that the partial bus-invert coding reduces the total bus transitions by 62.6% on the average, compared to that of the unencoded patterns.

5.3 Code Compression

An alternative approach to bus encoding is code compression. The basic strategy is to use narrow instruction codes for reducing the switching activity when the instructions are transmitted from a program memory to a CPU.

One of the best known instruction compression approaches is the "Thumb" instruction set of the ARM microprocessor family [Segars95]. ARM cores can be programmed using a reduced set of 16-bit instructions instead of standard 32-bit RISC instructions, which reduces required instruction memory occupation and bandwidth by a factor of 2.

Yoshida et al. proposed a code compression technique as depicted in Figure 6-21 [Yoshida97]. Suppose we have an object code and the number of distinct instructions appeared in the code is *N*. In this case, we can express all those instruction codes using $\lceil \log N \rceil$-bit binary patterns. Since the firmware running on a given embedded processor normally uses only a small subset of the instructions supported by the processor, a $\lceil \log N \rceil$-bit is much smaller than original instruction width. As a result, we can reduce the energy consumption for fetching instruction. According to this idea, the object code is stored in memory in compressed format, i.e., each instruction is replaced with a $\lceil \log N \rceil$-bit binary pattern which is in one-to-one correspondence with the original instruction. Every time an instruction is fetched from the program memory, it is decompressed (i.e., the original format is restored) using an *instruction decompression table* (IDT) and then passed to the processor's decoding logic. This architecture is motivated by the fact that software programs normally use only a subset of all possible instructions offered by the processor's instruction set. Since $\lceil \log N \rceil$ (where *N* is the number of distinct instructions) is usually much smaller than the original instruction width, this approach reduces both memory energy and bus power consumption.

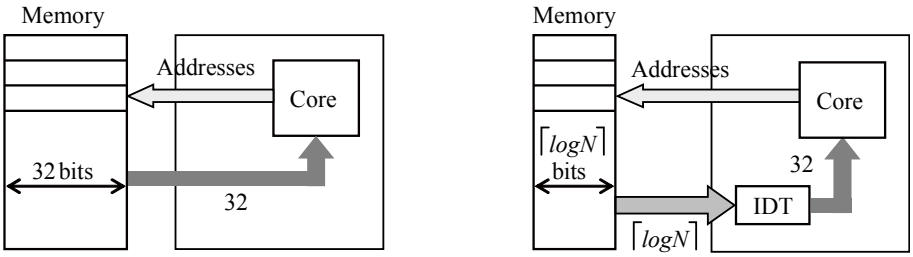


Figure 6-21. An Example of Code Compression

Although, in principle, the solution depicted above offers good opportunities for energy reduction, it often happens that the number of

distinct instructions, *N*, used by a program is not small. In such a situation, the size of the *Instruction Decompression Table* (IDT) becomes very large, and therefore area and power dissipation of the IDT would be very large as well. As a solution of the problem, Benini et al. proposed a selective instruction compressing technique [Benini99]. Their idea is to compress only a subset of fixed cardinality (256 elements) of the instructions used by a program, namely, those that are executed more often. This approach is motivated by the observation that the 256 most frequently used instructions are always executed for at least 50% and up to 99.99% of the time. The idea can be implemented as shown in Figure 6-22. This approach guarantees a fixed and limited size for the IDT and reduces energy and area overhead for decompressing the instructions.

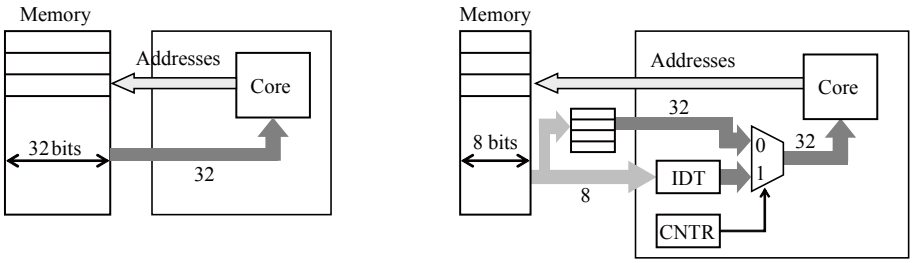


Figure 6-22. Selective Code Compression

5.4 Conclusions

We addressed several techniques for lowering switching activity of off-chip buses considering tradeoff between the power consumption for on-chip computation and that for off-chip communication. Other than the techniques addressed in this section, there has been proposed a lot of techniques which reduce switching activity of high capacitance nodes. Specifically, circuit level approaches like logic synthesis techniques, placing and routing techniques, and high-level synthesis techniques which reduce transitions of high capacitance modules are well studied. On the other hand, there is much scope left to study on source-level design techniques which modify an application program in a way that power-hungry hardware components are less frequently used without sacrificing performance, computational quality and system reliability.

6. TECHNIQUES FOR REDUCING REAKAGE POWER

For mobile/portable devices with a high standby-to-active ratio, leakage current may be the dominant factor in determining overall battery life. The three primary sources of leakage current (See Figure 6-23) are sub-threshold (I_{sub}) or source-to-drain leakage current which grows exponential with lowering V_t and increasing temperature, reverse bias junction band-to-band tunneling current ($I_{b\text{-}b}$), and gate oxide tunneling current (I_{gate}). Reducing of gate oxide thickness results in an increase in the field across the oxide. The high electric field coupled with low oxide thickness results in tunneling of electrons from substrate to gate and also from gate to substrate through the gate oxide, resulting in the gate oxide tunneling current. Most of the interests have focused on the leakage caused by sub-threshold current and gate oxide tunneling current in terms of system level leakage management. Due to the leakage mechanisms described above, leakage current increases dramatically in the scaled devices. Particularly, with reduction of threshold voltage to achieve high performance, leakage power becomes a significant component of the total power consumption in both active and standby modes of operation. Since in the sleep mode I_{gate} will likely be dominant, two approaches may be considered: (1) reduce the threshold voltage of the sleep device somewhat (e.g. 100mV) to minimize the delay penalty associated with an extra series device; this allows the use of smaller sleep devices to simultaneously reduce I_{gate}, dynamic power, and layout area while not penalizing standby mode leakage since $I_{sub} << I_{gate}$ or (2) incorporate a multi-T_{ox} process was proposed.

A key difference between the state dependence of I_{sub} and I_{gate} is that the magnitude of I_{sub} primarily depends of the number of on vs. off transistors in a stack, while I_{gate} also depends strongly on the position of the on/off transistors.

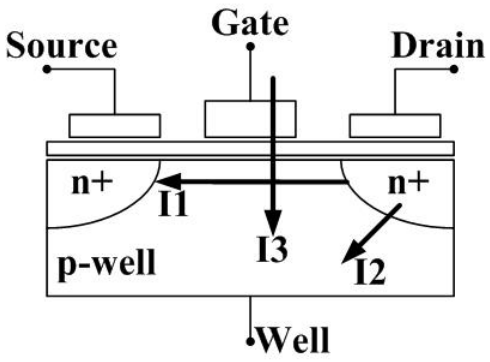


Figure 6-23. Sources of Leakage Current

Leakage power can be expressed as follows [6-2]:

$$P_{leak} = n \cdot I_{leak} \cdot V_{DD},\ I_{leak} \propto (V_{TH} / \alpha V_T)(1 - e^{-V_{DD}/V_{TH}}) \quad (6\text{-}1)$$

where n indicates the number of transistors, V_T denotes thermal voltage which is about 25mV at room temperature and increases linearly as temperature increases. According to this relationship, leakage current and therefore power dissipation increases exponentially with decreasing threshold voltage (V_{TH}) and with increasing temperature. Equation (6-1) suggests two ways to reduce P_{leak}. First, we could turn off the supply voltage. That is, set V_{DD} to zero so that the factor in parentheses also becomes zero. Second, we could increase the threshold voltage, which (because it appears as a negative exponent) can have a dramatic effect in even small increments. Of course using high-V_T transistors will degrade performance. A solution is to have mixture of high and low V_T transistors. Use low V_T transistors on timing-critical paths and high Vt transistors on non-critical paths. This approach is referred to as dual V_T design. Multi-Threshold CMOS (MTCMOS) cells can be used to control leakage power (See Figure 6-24). Low V_T transistors are used to implement gates for high speed, while high V_T transistors are added to form virtual rails. These high V_T transistors suppress the leakage current when the sleep signal is activated. Of course, there needs to be a sleep control mechanism.

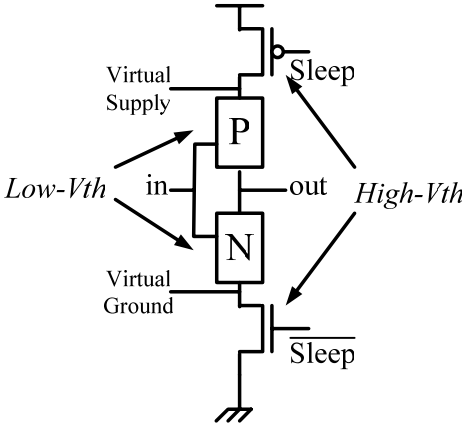


Figure 6-24. Multi-Threshold CMOS (MTCMOS)

Variable Threshold CMOS (VTCMOS) is a body biasing technique that controls effective threshold voltage by applying substrate bias to MOS transistors (See Figure 6-25). This technique is applicable at runtime. In the active mode, a zero body bias is applied. In standby mode, the effective threshold voltage is made to be larger by applying a reverse substrate bias to block the leakage current. Transistor performance in the active mode is kept the same as that in the conventional design by utilizing low V_{DD} and low V_T. However, triple well technology is required.

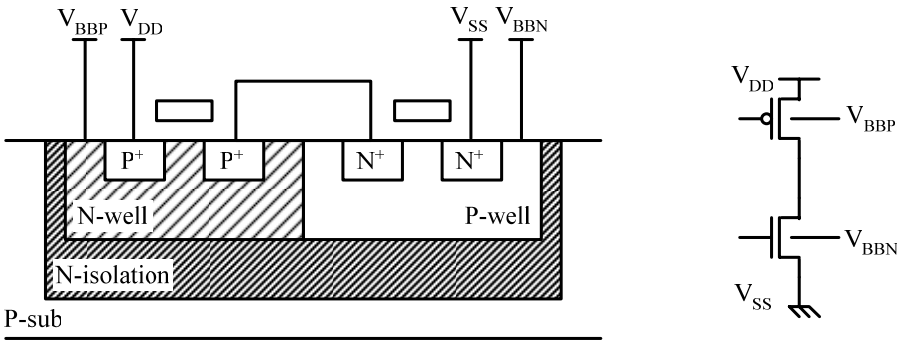


Figure 6-25. Variable-Threshold CMOS (VTCMOS)

In addition to above approaches, area reduction also reduces leakage power. Datapath width adjustment described in Section 6.4 is also effective for reducing the leakage power. The power dissipation of the whole system not only dynamic power but also leakage power is drastically reduced by

tuning the parameters of processors and memories tailored for the applications.

Reducing the number of transistors and controlling power supply voltage, V_T , or temperature dynamically can reduce the leakage. Basic strategies are shown below. Some system level methodologies related using the strategies are shown in this section.

- using high threshold voltage for non-critical paths
- shifting the circuit to the low leakage mode
- cooling high temperature parts,
- reducing the number of transistors.

Many techniques [Ishihara2002, Kaxiras2000, Powell2000, Sato2004] proposed to address leakage power have focused on cache memory that is a major leakage consumer of the entire system because leakage power is a function of the number of transistors. For example, StrongARM processor uses 60% of the die area for cache memories [Manne1998].

6.1 Multiple Vth CMOS and Dual Vth techniques

One way to increase the threshold voltage is to use Multiple Threshold Circuits with sleep transistors [Calhoun2003]. This involves isolating a leaky circuit element by connecting it to a pair of virtual power supplies that are linked to its actual power supplies through sleep transistors (Figure 6-24). When the circuit is active, the sleep transistors are activated, connecting the circuit to its power supplies. However, when the circuit is inactive, the sleep transistors are deactivated, thus disconnecting the circuit from its power supplies. In this inactive state, almost no leakage passes through the circuit because the sleep transistors have high threshold voltages. This technique effectively confines the leakage to one part of the circuit, but is tricky to implement for several reasons. The sleep transistors must be sized properly to minimize the overhead of activating them. They cannot be turned on and off too frequently. Moreover, this technique does not readily apply to memories, because memories lose data when their power supplies are cut.

Another way to increase the threshold is to employ dual threshold circuits. Dual threshold circuits [Liu2004, Wei1998, Ho2004] reduce leakage by using high threshold (low leakage) transistors on non-critical paths and leakage by using low threshold transistors on critical paths, the idea being that non-critical paths can execute instructions more slowly without impairing performance.

6.2 Dynamic Power Management for Reducing Leakage

Adaptive body biasing technique [Seta1995, Kobayashi1994,Nose2002] is a runtime technique that reduces leakage power by dynamically adjusting the threshold voltages of circuits depending on whether the circuits are active. When a circuit is not active, the technique increases its threshold voltage, thus saving leakage power exponentially, although at the expense of a delay in circuit operation. When the circuit is active, the technique decreases the threshold voltage to avoid slowing it down. To adjust the threshold voltage, adaptive body biasing applies a voltage to the transistor's body known as a body bias voltage (Figure 6-25). Vt is dynamically controlled through software depending on the workload of a processor. The Vth-hopping scheme [Nose2002] can achieve 82% power saving compared with the fixed low-Vth circuits. In order to suppress efficiently the leakage power, combining the adaptive body biasing technique and the dual Vt technique could be useful (See Figure 6-26). In this case, the adaptive body biasing is used only in the critical paths. On the other hand, Vt of the non-critical paths gates is set to a considerably higher value (high-Vt), which is not changed for the entire time.

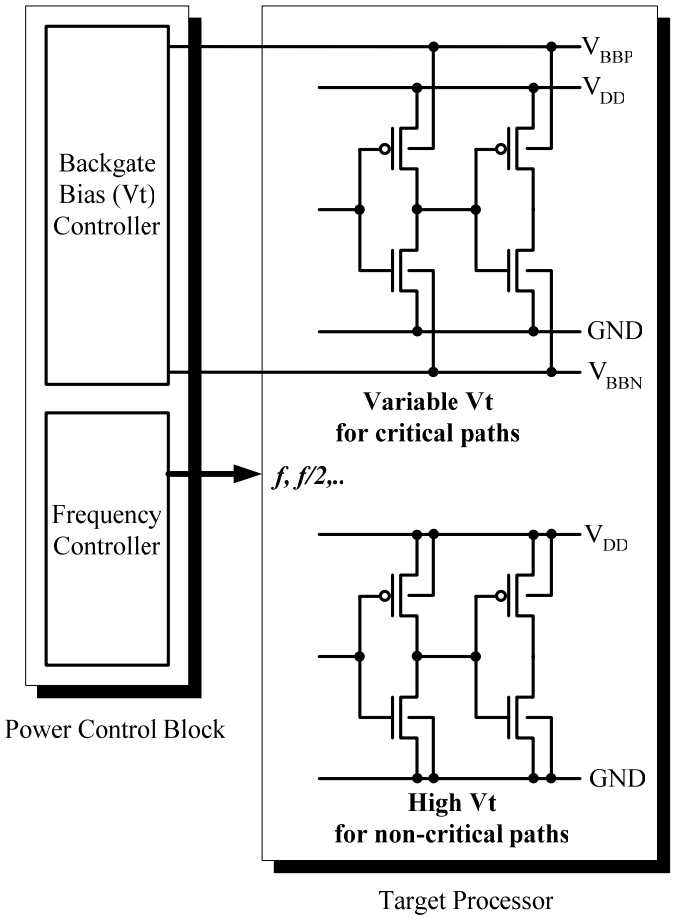


Figure 6-26. Combining VTCMOS and Dual Vth Technologies

6.3 Thermal Management

Several cooling techniques have been developed since the 1960s. Some below cold air into the circuit, while others refrigerate the processor [Schmidt2002], sometimes even by costly means such as circulating cryogenic fluids like liquid nitrogen [Krane1988]. These techniques have three advantages. First, they significantly reduce subthreshold leakage. In fact, a recent study [Schmidt2002] showed that cooling a memory cell by 50 degrees Celsius reduces the leakage power by five times. Second, these techniques allow a circuit to work faster because electricity encounters less resistance at lower temperatures. Third, cooling eliminates some negative effects of high temperatures, namely the degradation of a chip's reliability

and life expectancy. Recently, the reliability is a much more significant issue in design. Despite these advantages, there are issues to consider, such as the costs of the hardware used to cool the circuit. Moreover, cooling techniques are insufficient if they result in wide temperature variations in different parts of a circuit. Rather, one needs to prevent "hotspots" by distributing heat evenly throughout a chip.

Reliability and leakage power are both strongly affected by system temperature. In [Simunic], they proposed a joint reliability and power management optimization. Their approach achieved a significant improvement in energy consumption (40%) in tandem with meeting reliability constraint for all operating temperatures.

Another thermal management is a temperature aware task scheduling [Hung2005], which is task scheduling such that the temperature of HW is minimized.

6.4 Bitwidth Optimization for Reducing Leakage

Cao et. al. [Cao2002] reported a bitwidth optimization technique for reducing not only dynamic and leakage power at system level design. For Lempel-Ziv algorithm, they got dynamic power saving of 59.2% and leakage power saving of 64.3 a the optimal datapath width of 15bits; for ADPCM encoder, dynamic power saving is 44.2% and leakage power saving is 4.74% at the optimal datapath width of 19bits; for MPEG-2 AAC audio decoder, the dynamic power saving is 14.5% and leakage power saving is 18.1% at the optimal datapath width of 24bits and MPEG2 video decoder, the dynamic power saving is 18.3% and leakage power is 19.1% at optimal datapath width of 28bits. For different application, the number of variables is different and the effective size of variables is also different, therefore the optimal datapath width of minimal power is different. Note that this is under the assumption *ActTime* : *InactTime* = 1 : 1. *ActiTime* is the application execution time, which is called active time and *InactTime* is the idle time, which is called inactive time.

6.5 Commercial Products

In [Mutoh1996], they presented power management processor, which uses MTCMOS technology.

Toshiba used the mixed MTCMOS and Dual V_T method to reduce the leakage power in a DSP core for W-CDMA cell phones. Cell phones spend a significant amount of time in the standby mode. Toshiba also presented a low power single-chip MPEG4 video-phone LSI. The VTCMOS technology

is employed to reduce a standby leakage current, which is only 17% of the conventional CMOS design [ISSCC A 60MHz 240mW MPEG-4 video-phone LSI with 16Mbit embedded DRAM].

6.6 Conclusions

This section describes leakage power reduction methodologies. There are four basic strategies: using high-V_T on non-critical paths, shifting low leakage mode, cooling high temperature parts, and reducing the number of transistors.

7. POWER REDUCTION TECHNIQUES USING APPLICATION SPECIFIC HARDWARE

The ultimate way for energy reduction is creating application-specific integrated circuits (ASICs) that implement their algorithms directly in dedicated, fixed-function logic. The most energy-efficient type of processor core is the "application-specific instruction processor" (ASIP). These processors are custom designed for the application at hand. Today, however, a few companies offer automated tools that generate ASIPs based on parameters supplied by the system designer. ASIC designers can also achieve good energy efficiency by starting with a processor core and then customizing the core to the needs of their application. The processor cores offered by ARC and Tensilica are specifically designed for customization by the system designer. Both companies' offerings allow the system designer to add custom instructions that can produce massive energy efficiency gains.

7.1 Energy-Flexibility Tradeoff

Power consumption heavily depends on an implementation style and its flexibility [Rabaey00]. In Figure 6-27, the tradeoff between energy consumption and flexibility for different architectures is shown. As one can see, the dedicated hardware (ASICs) is 4 orders of magnitude more power efficient than embedded processors. Therefore, if there is no need for flexibility, the ASIC implementation is preferred. In practice, however, many systems require flexibility of the system in order to support not only existing applications but also upcoming ones.

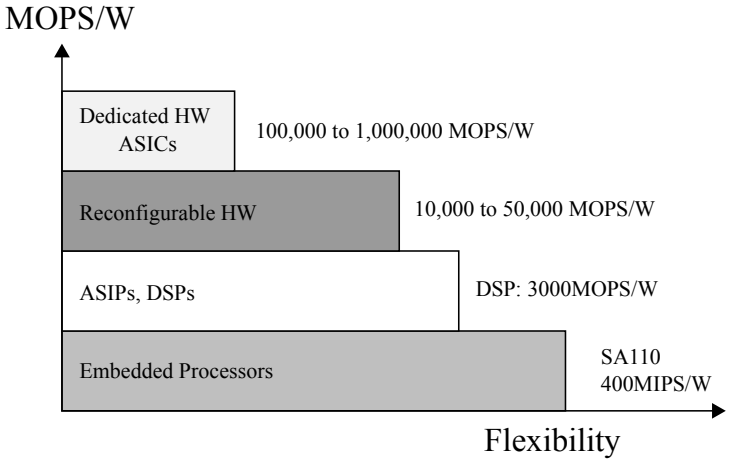


Figure 6-27. Energy-Flexibility Tradeoff

We can broadly categorize system architectures which concurrently satisfy high flexibility and low energy consumption as follows,

1. A hybrid architecture which consists of embedded processor or DSP and dedicated hardware, and
2. A configurable processor.

7.2 Hybrid Architecture

A hybrid-architecture consists of a microprocessor core, a set of standard cores, and a set of application specific cores as shown in Figure 6-28. The design goal using the hybrid-architecture is to partition a given application into the microprocessor core and the application specific cores in order to minimize the total energy consumption.

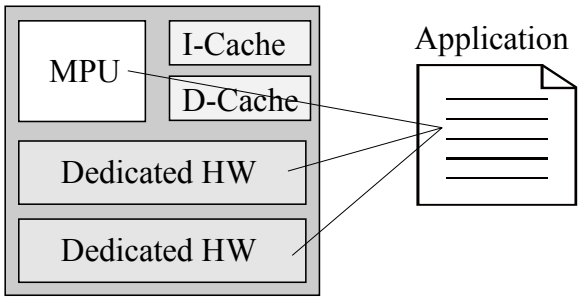


Figure 6-28. An Example of a Hybrid-Architecture

Hardware/software partitioning is the process of dividing an application into software running on a microprocessor and dedicated hardware. This approach is a well-established design methodology with the goal to increase the performance and to decrease the energy consumption of a system as described.

Dave et al. proposed a hardware/software co-design technique, called COSYN, which targets embedded systems consisting of general-purpose processors, ASICs and FPGAs [Dave97]. Functions of COSYN include allocation, scheduling, performance estimation, and power optimization. COSYN finds hardware/software partitioning based on the performance and power estimation of a processing element.

Henkel proposed a hardware/software partitioning technique for low-power core-based systems [Henkel99]. The technique considers the power consumption of a whole embedded system consisting of a microprocessor core, application specific cores, cache cores and a memory core. The technique based on a fine-grained (instruction/operation-level) analysis of energy consumption. The experimental results demonstrated high reductions of power consumption between 35% and 94% at the cost of a relatively small additional hardware overhead.

7.3 Configurable Processor

A configurable processor core is a fully functional processor design that can be customized or expanded to meet the performance and/or energy efficiency needs of applications [Wei05]. There are four general ways a processor can be configured:

- By selecting from standard configuration options, such as bus widths, interfaces, memories, floating-point units, etc.
- By adding custom instructions that describe new registers, register files and custom data types, such as 56-bit data for security processing or 256-bit data types for packet processing.
- By adding custom, high-performance interfaces that exceed the bandwidth abilities of the more common shared-bus architectures of conventional RISC and DSP cores.

Configurable processors are typically delivered as synthesizable RTL code, and can be easily mapped onto an FPGA or SoC design. Some configurable processors are provided with automatically tailored software-development tools (the compiler, assembler, debugger, linker, and profiler), EDA synthesis scripts, and verification test benches that reflect the designer-

defined architectural extensions so that no additional effort is required to ready the configured core for SoC development.

The ability to add custom instructions of any width allows an SoC designer to use a configurable processor core to implement datapath operations that closely match the abilities of a manually designed RTL block. Since the configurable processor does not have a feature for dynamically reconfiguring the structure of the processor, it is more energy efficient than a reconfigurable processor. In the configurable processor core, the datapaths are implemented using the base processor's integer pipeline, plus additional execution units, registers, and other functions added by the chip architect or SoC designer for a target application.

Energy efficiency of the configurable processor typically comes from the following three features [Wei05],

1. Configuration of the instruction set permits a much closer fit of the processor to the target application,
2. Configuring the processor removes unneeded hardware features like larger cache memories than needed, unused register files and extra bits of datapath [Inoue00], and
3. Automatic processor generation tools enable logic optimization, signal switching activity reduction, and seamless mapping into low-voltage circuits.

A lot of configurable processors and their optimization methodologies are proposed. However, only a few of them focus on methodologies for lowering energy consumption.

In [Inoue00], Inoue et al. proposed a flexible SoC architecture and its optimization framework, called FlexSys, which allows system designers to customize datapath width and memory size for a target application. A key of the FlexSys technology is that it allows designers to customize the core processor for the target application by replacing a few photomasks used for via layers only, which results in a low-cost customization of the processor for a target application. The experiments using DSPstone benchmark programs demonstrated that the energy consumption can be reduced by 54% compared to the normal RISC processor-based system which has a CPU core with 32-bit datapath and the fixed number of memory words.

7.4 Conclusions

In this section we introduced a concept of energy-flexibility tradeoff. We showed that system designers can drastically reduce energy consumption by

trading flexibility for energy consumption. However, in practice, it is very important to preserve system flexibility in case of future upgrade or modification in a target application. Therefore, we have to find the best compromising point between high flexibility and low energy consumption. We can broadly categorize system-level methodologies which satisfy high flexibility with low energy consumption as follows,

1. hardware/software partitioning for a hybrid architecture which consists of a microprocessor core and dedicated hardware and
2. exploiting customizability of configurable processors.

These strategies allow system designers to explore SoC architectures considering tradeoff between flexibility and energy consumption. As a result, system designers can find the best tradeoff point which compromises between high flexibility and low energy consumption.

8. SUMMARY

This chapter addressed several key methodologies for reducing power and/or energy consumption of SoCs which consist of hardware and software running on it. Each of those methodologies takes design tradeoffs into consideration. In Section 3, we introduced an energy-delay tradeoff and an energy-reliability tradeoff in SoC design. Section 4 discussed on a tradeoff between energy consumption and quality of services (QoS). The QoS, in this chapter includes precision (or computational quality) and latency (or response time). In section 5, a tradeoff between computational energy and communication energy is considered. Section 6 summarized several leakage reduction techniques considering the energy-delay tradeoff and the energy-QoS tradeoff. In Section 7, we introduced an energy-flexibility tradeoff. The key point of the energy reduction techniques is to take the tradeoffs into consideration according to a design objective and design constraints.

The problem of how to model and evaluate complicated SoCs in terms of energy, performance, QoS, reliability and flexibility becomes more attractive to tackle. As the supply voltage and threshold voltage of chips is lowered down along with the transistor scaling, sensitivity to temperature variation, process variation, sources of soft error and noise sources is increased. This results in model uncertainty and makes evaluation of SoC difficult. Increasing size, complexity, and functionality integrated on SoC becomes this problem more difficult. In near future, modeling and evaluation of SoC dynamically and/or statically taking the model uncertainty into account is one of the most important themes for low-energy SoC design.

REFERENCE

[Black69] J. R. Black, "Electromigration Failure Modes in Aluminum Metallization for Semiconductor Devices, " in Proc. of IEEE, vol. 57, no. 9, pp.1587-1594, Sep. 1969.

[Weste93] N. Weste and K. Eshraghian, "Principles of CMOS VLSI design", Addison-Wesley, 1993.

[Chatterjee96] A. Chatterjee, M. Nandakumar, and I. Chen, "An Investigation of the Impact of Technology Scaling on Power Wasted as Short-Circuit Current in Low Voltage Static CMOS Circuits," in Proc. ISLPED, pp.145-150, Aug. 1996.

[Usami95] K. Usami, and M. Horowitz, "Clustered Voltage Scaling Techniue for Low-Power Design", in Proc. of Int'l Symposium on Low Power Design, pp.3-8, April, 1995.

[Johnson97] M. C. Johnson and K. Roy, "Datapath Scheduling with Multiple Supply Voltages and Level Converters," ACM TODAES, vol.2, no.3, pp.227-248, July, 1997.

[Chandrakasan95] A. P. Chandrakasan, M. Potkonjak, R. Mehra, J. Rabaey, and R. W. Brodersen, "Optimizing Power Using Transformations," IEEE Trans. on CAD, vol.14, no.1, pp.12-31, Jan., 1995.

[Raghunathan94] A. Raghunathan and H. K. Jha, "Behavioral Synthesis for Low Power", in Proc. of Int'l Conference on Computer Design, pp.318-322, Oct., 1994.

[Raghunathan95] A. Raghunathan and H. K. Jha, "An Iterative Improvement Algorithm for Low Power Data Path Synthesis", in Proc. of Int'l Conference on Computer Aided Design, pp.597-602, Nov., 1995.

[Goodby94] L. Goodby, A. Orailoglu, and P. M. Chau, "Microarchitectural Synthesis of Performance-Constrained Low-Power VLSI Designs", In Proc. of Int'l Conference on Computer Design, pp.323-326, Oct., 1994.

[Kumar95] N. Kumar, S. Katkoori, L. Rader, and R. Vemuri, "Profile-Driven Behavioral Synthesis for Low-Power VLSI Systems", IEEE Design & Test, vol.12, no.3, pp.70-84, Fall, 1995.

[Martin95] R. S. Martin, and J. P. Knight, "Power-Profiler: Optimizing ASICs Power Consumption at the Behavioral Level", In Proc. of Design Automation Conference, pp.42-47, June, 1995.

[Raje95] S. Raje, and M. Sarrafzadeh, "Variable Voltage Scheduling", in Proc. of Int'l Symposium on Low Power Design, pp.9-14, April, 1995.

[Lin97] Y. R. Lin, C. T. Hwang and A. C.-H. Wu, "Scheduling Techniques for Variable Voltage Low Power Designs", ACM TODAES, vol.2, no.2, pp.81-97, April, 1997.

[Chang96] J. Chang and M. Pedram, "Energy Minimization Using Multiple Supply Voltages", in Proc. of Int'l Symposium on Low Power Electronics and Design, pp.157-162, Aug., 1996.

[Ishihara98] T. Ishihara, and H. Yasuura, "Voltage Scheduling Problem for Dynamically Variable Voltage Processors", in Proc. of Int'l Symposium on Low Power Electronics and Design, pp. 197-202, Aug. 1998.

[Weiser94] M. Weiser, B. Welch, A. Demers and S. Shenker, "Scheduling for Reduced CPU Energy", in Proc. of Symposium on Operating Systems Design and Impremention, pp.13-23, Nov., 1994.

[Yao95] F. Yao, A. Demers and S. Shenker, "A Scheduling Model for Reduced CPU Energy", in Proc. of Symposium on Faundations of Cumputer Science, pp. 374-382, Oct., 1995.

[Shin99] Y. Shin and K. Choi, "Power Conscious Fixed Priority Scheduling for Hard Real-Time Systems", in Proc. of Design Automation Conference, pp.134-139, June, 1999.

[Shin00] Y. Shin, K. Choi and T. Sakurai, "Power Optimization of Real-Time Embedded Systems on Variable Speed Processors", in Proc. of Int'l Conference on Computer Aided Design, pp.365-368, Nov., 2000.

[Okuma99] T. Okuma, T. Ishihara, and H. Yasuura, "Real-Time Task Scheduling for a Variable Voltage Processor", in Proc. of Int'l Symposium on System Synthesis, pp.24-29, Nov., 1999.

[Austin04] T. Austin, D. Blaauw, T. Mudge and K. Flautner, "Making Typical Silicon Matter with Razor", IEEE Computer Magazien, pp.57-65, March 2004.

[Bertozzi02] D. Bertozzi, L. Benini and G. De Micheli, "Low-Power Error-Resilient Encoding for On-Chip Data Busses", in Proc of Dasign Automation and Test in Europe Conference, pp.102-109, March, 2002.

[Worm02] F. Worm, P. Lenne, P. Thiran and G. De Micheli, "An adaptive low-power transmission scheme for on-chip networks", Proc. of Int'l symposium on system synthesis, pp.92-100, Oct. 2002

[Cao2003] Y. Cao, and H. Yasuura, "Quality-Driven Design by Bitwidth Optimization for Video Applications," in Proc. Asia and South Pacific Design Automation Conference, pp., 2003.

[Alalusi2000] S. Alalusi, and B. Victor, "Variable Word Width Computation for Low Power," CS 252 Computer Architecture, 2000.

[Sstephenson2000] M. Stephenson, J. Babb, and S. Amarasinghe, "Bitwidth Analysis with Application to Silicon Compilation," in Proc. ACM SIGPLAN 2000 Conference on Programming language design and implementation, pp.198-120, 2000.

[Canal2000] R. Canal, A. Gonzalez, and J. E. Smith, "Very Low Power Pipelines using Significance Compression," in Proc. of International Symposium on Microarchitecture, pp.181-190, 2000.

[Brooks2000] D. Brooks, and M. Martonosi, "Value-Based Clock Gating and Operation Packing: Dynamic Strategies for Improving Processor Power and Performance," in ACM Transactions on Computer Systems, vol. 18, no. 2, pp.89-126, May 2000.

[Bhunia2003] H. Li, S. Bhunia, Y. Chen, T. N. Vijaykumar, and K. Roy, "Deterministic Clock Gating for Microprocessor Power Reduction," in Proc. of International Symposium on High-Performance Computer Architecture, pp.113, 2003.

[Bellas1999] N. Bellas, I. Haji, and C. Polychronopoulos, "Using Dynamic Cache Management Techniques to Reduce Energy in a High-Performance Processor," in Proc. of International Symposium on Low Power Electronics and Design, pp.64-69, 1999.

[Benini1998] L. Benini, A. Bogliolo, S. Cavallucci, and B. Ricco, "Monitoring Systems Activity or OS-Directed Dynamic Power Management," in Proc. of International Symposium on Low Power Electronics and Design, pp.185-190, 1998.

[Wong2004] J. L. Wong, G. Qu, and M. Potkonjak, "Power Minimization in QoS Sensitive Systems," in IEEE Transactions on Very Large Scale Integration Systems, vol. 12, no. 6, pp.553-561, 2004.

[Qiu2001] Q. Qiu, Q. Wu, and M. Pedram, "Dynamic Power Management in a Mobile Multimedia System with Guaranteed Quality-of-Service," in Proc of Design Automation Conference, pp.834-839, 2001.

[Yardi2005] S. M. Yardi, M. S. Hsiao, T. L. Martin, and D. S. Ha, "Quality-Driven Proactive Computation Elimination for Power-Aware Multimedia Processing," in Proc of DATE, pp.340-345 , 2005.

[Pokam2004] G. Pokam, O. Rochecouste, A. Seznec, and F. Bodin, "Speculative Software Management of Datapath-width for Energy Optimization," in Proc. of LCTES, pp.78-87, 2004.

[Sinha2002] A. Sinha, A. Wang, and A. Chandrakasan,"Energy Scalable System Design," in IEEE Transactions on Very Large Scale Integration Systems, vol. 10, no. 2, 2002.

[Bellosa1999] F. Bellosa, "OS-Directed Throttling of Processor Activity for Dynamic Power Management," Tech. Report, 1999.

[Muroyama2003] M. Muroyama, A. Hyodo, T. Okuma, and H. Yasuura, "A Power Reduction Scheme for Data Buses by Dynamic Detection of Active Bits," in Proc. of Euromicro Symposium on Digital System Design - Architectures, Methods and Tools -, pp.408-415, 2003.

[Okuma2002] T. Okuma, Y. Cao, M. Muroyama, and H. Yasuura, "Reducing Access Energy of On-Chip Data Memory Considering Active Data Bitwidth," in Proc. of International Symposium on Low Power Electronics and Design, pp.88-91, 2002.

[Xanthopoulos2000] T. Xanthopoulos, and A. P. Chandrakasan, "A Low-Power DCT Core Using Adaptive Bitwidth and Arithmetic Activity Exploiting Signal Correlations and Quantization," in IEEE Jounal of Solid-State Circuits, vol. 5, no. 5, 2000.

[Liu94] D. Liu and C. Svensson, "Power Consumption Estimation in CMOS VLSI Chips," IEEE Journal of Solid-State Circuits, vol.29, no.6, pp.663-670, June 1994.

[Ko98] U. Ko, P. T. Balsara, and A. K. Nanda, "Energy Optimization of Multilevel Cache Architectures for RISC and CISC Processors," IEEE Trans. on VLSI Systems, vol.6, no.2, pp.299-308, June 1998.

[Su95] C. L. Su and A. M. Despain, "Cache Design Trade-offs for Power and Performance Optimization: A Case Study", In Proc. of ISLPED, pp.63-68, August 1995.

[Hicks97] P. Hicks, M. Walnock, and R. M. Owens, "Analysis of Power Consumption in Memory Hierarchies", In Proc. of ISLPED, pp.239-242, August 1997.

[Li98] Y. Li, and J. Henkel, "A Framework for Estimating and Minimizing Energy Dissipation of Embedded HW/SW Systems", In Proc. of DAC, pp.188-193, June, 1998.

[Shine99] W. T. Shine, and C. Chacrabarti, "Memory Exploration for Low Power, Embedded Systems", In Proc. of DAC, pp.140-145, June, 1999.

[Malik00] A. Malik, B. Moyer and D. Cermak, "A Low Power Unified Cache Architecture Providing Power and Performance Flexibility", In Proc. of ISLPED, pp.241-243, July 2000.

[Ishihara05] T. Ishihara and F. Fallah, "A Non-Uniform Cache Architecture for Low Power System Design", in Proc. of ISLPED, pp.363-368, Aug., 2005.

[McFarling89] S. McFarling, "Program Optimization for Instruction Caches", In Proc. of Int'l Conference on Architecture Support for Programming Languages and Operating Systems, pp.183-191, April 1989.

[Hwu89] W. W. Hwu and P. P. Chang, "Achieving High Instruction Cache Performance with an Optimizing Compiler", In Proc. of ISCA, pp.242-251, May 1989.

[Tomiyama96] H. Tomiyama and H. Yasuura, "Optimal Code Placement of Embedded Software for Instruction Caches", In Proc. of European Design and Test Conference, pp.96-101, March, 1996.

[Panda96] P. Panda, N. Dutt, and A. Nicolau, "Memory Organization for Improved Data Cache Performance in Embedded Processors", In Proc. of the 9th Int'l Symposium on System Synthesis, pp.90-95, November 1996.

[Hashemi97] A. H. Hashemi, D. R. Kaeli, and B. Calder, "Efficient Procedure Mapping Using Cache Line Coloring", in Proc. of Programming Language Design and Implementation, pp.171-182, June, 1997.

[Ghosh99] S. Ghosh, M. Martonosi, and S. Malik, "Cache Miss Equations: A Compiler Framework for Analyzing and Tuning Memory Behavior", ACM Trans. on Programming Languages and Systems, vol.21, no.4, pp.703-746, July, 1999.

[Kulkarni01] C. Kulkarni, C. Ghez, M. Miranda, F. Catthoor, H. De Man, "Cache Conscious Data Layout Organization for Conflict Miss Reduction in Embedded Multimedia Applications," Proc. of DATE 2001, pp.686-691, March 2001.

[Bankar02] R. Banakar, S. Steinke, B.-S. Lee, M. Balakrishnan, and P. Marwedel, "Scratchpad Memory : A Design Alternative for Cache On-Chip Memory in Embedded Systems", in Proc. of CODES, pp.73-78, May, 2002.

[Stan95] M. R. Stan and W. P. Burleson, "Bus-Invert Coding for Low-Power I/O," IEEE Trans. on VLSI Systems, vol. 3, pp.49-58, March, 1995.

[Su94] C. L. Su, C. Y. Tsui, and A. M. Despain, "Low Power Architecture Design and Compilation Technique for High-Performance Processors," in Proc. IEEE COMPCON, pp.209-214, Feb., 1994.

[Benini97] L. Benini, G. De Micheli, E. Macii, D. Sciuto, and C. Silvano, "Asymptotic Zero-Transition Activity Encoding for Address Busses in Low-Power Microprocessor-Based Systems," in Proc. of Great Lakes Symposium on VLSI, pp.77-82, March, 1997.

[Benini97-2] L. Benini, G. De Micheli, E. Macii, M. Poncino, and S. Quer, "System-Level Power Optimization of Special Pupose Applications: The Beach Solution," in Proc. of Int'l Symposium on Low Power Electronics and Design, pp. 24-29, August, 1997.

[Shin98] Y. Shin, S. Chae, and K. Choi, "Partial Bus-Invert Coding for Power Optimization of System Level Bus," in Proc. of Int'l Symposium on Low Power Electronics and Design, pp.127-129, August, 1998.

[Benini00] L. Benini, A. Macii, E. Macii, and M. Poncino, "Synthesis of Application Specific Memories for Power Optimization in Embedded Systems", in Proc. of Design Automation Conference, pp.300-303, June, 2000.

[Ishihara00] T. Ishihara, and H. Yasuura, "A Power Reduction Technique with Object Code Merging for Application Specific Embedded Processors", in Proc. of Design Automation and Test in Europe Conference, pp.617-623, March, 2000.

[Segars95] S. Segars, K. Clarke, L. Goudge, "Embedded Control Problems, Thumb and the ARM7TDMI," IEEE Micro, vol.15, no.5, pp.22-30, Oct., 1995.

[Yoshida97] Y. Yoshida, B. Y. Song, H. Okuhata, T. Onoye, and I. Shirakawa, "An Object Code Compression Approach to Embedded Processors," in Proc. of Int'l Symposium on Low Power Electronics and Design, pp.285-288, August, 1997.

[Benini99] L. Benini, A. Macii, E. Macii, and M. Poncino, "Selective Instruction Compression for Memory Energy Reduction in Embedded Systems", in Proc. of Int'l Symposium on Low Power Electronics and Design, pp.206-211, August, 1997.

[Powell2000] M. D. Powell, S. Yang, B. Falsafi, K. Roy, T. N. Vijaykumar, "Gated-Vdd: A Circuit Technique to Reduce Leakage in Deep-Submicron Cache Memories," in Proc. of International Symposium on Low Power Electronics and Design, pp.90-95, 2000.

[Yan2005] L. Yan, J. Luo, and N. K. Jha, "Joint Dynamic Voltage Scaling and Adaptive Body Biasing for Heterogeneous Distributed Real-Time Embedded Systems," IEEE Trans. on CAD, vol.24, no.7, pp.1030-1041, July 2005.

[Tsai2004] Y.-F. Tsai, D. E. Duarte, N. Vijaykrishnan, and M. J. Irwin, "Characterization and Modeling of Run-Time Techniques for Leakage Power Reduction," in IEEE Transactions on Very Large Scale Integration Systems, vol. 12, no. 11, 2004.

[Cao2002] Y. Cao, and H. Yasuura,"Leakage Power Reduction Using Bitwidth Optimization, " In Prof. of the 6th World Multiconference on Systemics, Cybernetics and Informatics., 2002.

[Li2005] P. Li, Y. Deng, and L. T. Pileggi, "Temperature-Dependent Optimization of Cache Leakage Power Dissipation," in Proc. of ICCD, 2005.

[Khouri2002] "Leakage Aware Synthesis," in Proc. of TVLSI 2002. x

[Ynag2001] P. Yang, C. Wung,..."Energy-Aware Runtime Scheduling for Embedded Multiprocessor SoCs," in IEEE Design and Test of Computers, 2001. x

[Ishihara2002] T. Ishihara, and K. Asada, "An architectural level energy reduction technique for deep-submicron cache memories," in Proc. of Asia and South Pacific Design Automation Conference, 2002.

[Kaxiras2000] S. Kaxiras, Z. Hu, G. Narlikar, and R. McLellan, "Cache-line decay: a mechanism to reduce cache leakage power," in Proc. of Workshop on Power Aware Computer Systems, 2000.

[Manne1998] S. Manne, A. Klauser, and D. Grunwald,"Pipeline gating: speculation control for energy reduction," in Proc. of International Symposium on Computer Architecture, 1998.

[Sato2004] H. Sato, and T. Sato, "A Static and Dynamic Energy Reduction Technique for I-Cache and BTB in Embedded Processors," in Proc. of Asia South Design Automation Conference, 2004.

[Schmidt2002] R. Schmidt, and B. Notohardjono, "High-end Server Low-Temperature Cooling," in IBM Journal of Research and Development, pp.739-751, 2002.

[Krane1988] R. Krane, J. Parsons, and A. Bar-Cohen, "Design of a candidate thermal control system for a cryogenically cooled computer," in IEEE Transactions on Components, Hybrids, and Manufacturing Technology, vol. 11, no. 4, pp.545-556, 1988.

[Calhoun2003] B. H. Calhoun, F. A. Honore, and A. Chandrakasan, "Design methodology for fine-grained leakage control in MTCMOS," In Proc. of International Symposium on Low Power Electronics and Design, pp.104-109, 2003.

[Liu2004] M. Liu, W.-S. Wang, and M. Orshansky, "Leakage Power Reduction by Dual-Vth Designs under Probabilistic Analysis of Vth Variation," In Proc. of International Symposium on Low Power Electronics and Design, pp.2-7, 2004

[Wei1998] L. Wei, Z. Chen, M. Johnson, K. Roy, and V. De, "Design and Optimization of Low Voltage High Performance Dual Threshold CMOS Circuits," in Proc. of Design Automation Conference, pp.489-494.

[Ho2004] Y.-T Ho, and T.-T. Hwang, "Low Power Design using Dual Threshold Voltage," in Proc. of Asia and South Pacific Design Automation Conference, pp.205-208, 2004.

[Seta1995] K. Seta, H. Hara, T. Kuroda, M. Kakumu, and T. Sakurai, "50% Active-Power Saving without Speed Degradation using Standby Power Reduction (SPR) circuit," in Proc. of ISSCC, pp.318-319, 1995.

[Kobayashi1994] Kobayashi, and T. Sakurai, "Self-Adjusting Threshold Voltage Scheme (SATS) for Low-Voltage High-Speed Operation," in Proc. of CICC, pp.271-274, 1994.

[Claasen] T. Claasen,... "" in Proc. of ISSCC99.

[Budiu2002] M. Budiu, "Application-Specific Hardware," in Proc. International Conference on Field Programmable Logic and Applications, pp.853-863, 2002.

[Ranpara1999] S. Ranpara et al., "A Low-Power Viterbi Decoder Design for Wireless Communications Applications," Proc. ASIC, 1999.

[Gilbert2001] F. Gilbert et al. "Low Power Implementation of a Turbo-Decoder on Programmable Architectures" in Proc. ASP-DAC, 2001.

[Usami] K. Usami et al. "Design Methodology of Ultra Low-power MPEG4 Codec Core Exploiting Voltage Scaling Techniques"

[Lee2003] R. B. Lee, "Challenges in the Design of Security-Aware Processors," in Proc. Application-Specific Systems, Architectures, and Processors, pp. , 2003.

[Udayakumaran2002] S. Udayakumaran, B. Narahari, R. Simha, "Application-Specific Memory Partitioning for Low Power Consumption," COLP 2002.

[Rabaey00] J. M. Rabaey, “Low Power Silicon Architecture for Wireless Communications,” in Proc. of Asia South Pacific Design Automation Conference, pp.377-380, January, 2000.

[Wei05] J. Wei, and C. Rowen, “Implementing Low-Power Configurable Processors – Practical Options and Tradeoffs,” in Proc. of Design Automation Conference, pp.706-711, June, 2005.

[Dave97] B. P. Dave, G. Lakshminarayana, and N. K. Jha, “COSYN: Hardware-Software Co-Synthesis of Embedded Systems,” in Proc. of Design Automation Conference, pp.703-708, June, 1997.

[Henkel99] J. Henkel, “A Low Power Hardware/Software Partitioning Approach for Core-based Embedded Systems,” in Proc. of Design Automation Conference, pp.122-127, June, 1999.

[Inoue00] A. Inoue, T. Ishihara, and H. Yasuura, ``Flexible System LSI for Embedded Systems and Its Optimization Techniques", In H. Yasuura, editors, Journal of Design Automation for Embedded Systems, Vol.5, No.2, pp.179-205, Kluwer Academic Publishers, Jun. 2000.