A HARDWARE-SOFTWARE PROCESSOR ARCHITECTURE USING

PIPELINE STALLS FOR LEAKAGE POWER MANAGEMENT

Except where reference is made to the work of others, the work described in this thesis is my own or was done in collaboration with my advisory committee. This thesis does not include proprietary or classified information.

Khushboo Sheth

Certificate of Approval:

Adit D. Singh James B. Davis Professor Electrical and Computer Engineering	Vishwani D. Agrawal, Chair James J. Danaher Professor Electrical and Computer Engineering
Victor P. Nelson Professor Electrical and Computer Engineering	George T. Flowers Dean Graduate School

A HARDWARE-SOFTWARE PROCESSOR ARCHITECTURE USING PIPELINE STALLS FOR LEAKAGE POWER MANAGEMENT

Khushboo Sheth

A Thesis

Submitted to

the Graduate Faculty of

Auburn University

in Partial Fulfillment of the

Requirements for the

Degree of

Master of Science

Auburn, Alabama
May 9th, 2009

A HARDWARE-SOFTWARE PROCESSOR ARCHITECTURE USING PIPELINE STALLS FOR LEAKAGE POWER MANAGEMENT

Khushboo Sheth

Signature of Author

Date of Graduation

Vita

Khushboo U. Sheth, daughter of Mr. Umeshkumar R. Sheth and Mrs. Nisha U. Sheth, was born in Ahmedabad, Gujarat, India. She attended L. D. College Of Engineering, Gujarat University in 2001 and graduated with Bachelor of Engineering degree in Instrumentation and Control Engineering in 2005. At the same year August, she entered the Electrical and Computer Engineering graduate program at Auburn University, Alabama.

Thesis Abstract

A HARDWARE-SOFTWARE PROCESSOR ARCHITECTURE USING PIPELINE STALLS FOR LEAKAGE POWER MANAGEMENT

Khushboo Sheth

Master of Science, May 9th, 2009
(B.E., Gujarat University, 2005)

109 Typed Pages

Directed by Vishwani D. Agrawal

In recent years, power consumption has become a critical design concern for many VLSI systems. Nowhere is this truer than for portable, battery-operated applications, where power consumption has perhaps superseded speed and area as the overriding implementation constraint. But since last few years as the greater emphasis is put on miniaturization, in future technologies, the problem of subthreshold leakage power in CMOS circuits will grow in significance. The leakage current is exponentially dependent on the value of the threshold voltage such that if the threshold voltage is reduced (as it will be in the future technologies), the leakage current registers an exponential increase. Responding to this challenge, several low power techniques at levels ranging from technology to architecture have been proposed to reduce both dynamic and static power for processors and make them more energy-efficient. Some of these techniques can be applied to hardware whereas others are software based techniques. In this thesis, we propose a combined hardware-software technique which will potentially show considerable leakage energy reduction when power-performance trade-offs are made in higher-leakage technologies.

A simple method to reduce the power consumption in a processor is to slow down the clock frequency. The dynamic power reduces in proportion to the frequeny reduction. However, the leakage power remains the same. Because a computing task will require the same number of clock cycles it will now take more time. The leakage energy will therefore increase, although the dynamic energy will remain the same. It is the reduction of leakage power and energy that is targeted in the present work.

The main idea introduced and investigated is that the power-performance trade-off is accomplished by inserting empty (no-op) cycles while the clock rate is kept unchanged. The hardware units are especially designed to save leakage power while processing a no-op instruction. As an illustration, the hardware of a five-stage pipeline RISC processor is redesigned to reduce power consumption of the no-op cycles using sleep modes. This largely eliminates the leakage in those cycles. The expected result is that as more empty cycles are inserted, the performance would drop similar to the conventional clock frequency reduction. However, the computing task that now takes more cycles has only a marginal increase in the leakage energy. In addition, the empty cycles may eliminate many of the pipeline hazards and thus reduce the performance penalty. In this work, power supply for the active (i.e., non-empty) cycles is not changed and that aspect is left for the future investigation.

The control unit of the processor has been designed to interpret an external power management signal. Based on the power and performance requirements, this signal specifies a performance slowdown factor. The power block has been added in the processor architecture that inserts no-ops in proportion to the slowdown factor in the instruction stream. The normal clock rate is maintained. The control also generates the power signals for different blocks of the processor along with the other control signals. These power signals are

applied to various hardware blocks (register file, ALU, and instruction and data caches) of the processor, which on the basis of the required activity are put into one of the low power modes such as drowsy or sleep mode. These modes are chosen for the best leakage power reduction.

We have simulated a modified 32-bit MIPS pipelined processor for 22nm and 65nm technologies using the Berkeley Predictive Technology Models.

Acknowledgments

I thank my adviser Dr. Vishwani D. Agrawal for his constant support. Without his patient guidance and encouragement, this work would not be possible. His technical advice made my master studies a meaningful learning experience. I thank my advisory committee members, Dr. Victor P. Nelson and Dr. Adit D. Singh for being on my thesis committee and for their invaluable advice on this research. I thank all my professors in Auburn University for their guidance and support, its been a privilege learning from them. I would like to acknowledge with gratitude and affection, encouragement and support given by my parents and my brother during my graduate study. I thank my research mates and friends in Auburn for their assistance and company.

I acknowledge with thanks a teaching assistantship at Auburn, which allowed me to take a closer look at microprocessors. My work was additonally supported by the NSF Grant 421128.

Style manual or journal used Journal of Approximation Theory (together with the style known as "aums"). Bibliograpy follows van Leunen's *A Handbook for Scholars*.

Computer software used The document preparation package TEX (specifically LATEX) together with the departmental style-file `aums.sty`.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The invention of a transistor was a giant leap forward for low-power electronics that has remained unequaled to date. The operation of vacuum tubes required several hundred volts and several watts of power. In comparison the transistor required only milliwatts of power. Since the invention of the transistor, decades ago, through the years leading to the 21st century, power dissipation, though not entirely ignored, was of little concern. The greater emphasis was on performance and miniaturization. Applications powered by batteries - pocket calculators, hearing aids and, most importantly, wristwatches - drove low-power electronics. In all such applications, it is important to prolong the battery life as much as possible. Especially now, with the growing trend towards portable computing and wireless communication, power dissipation has become one of the most critical factors in the continued development of the microelectronics technology. There are two reasons for this:

1. One of the most significant low-power design ideas of the last century, the complementary metal oxide semiconductor (CMOS) technology [111], has served us well for several decades. However, to make the best cost-performance trade-off [82] we continue to integrate more functions onto a chip by shrinking the features to the smallest size permitted by the manufacturing technology. As a result, the dissipation of power per unit area grows and the accompanying problem of heat removal and cooling worsens. Examples are the general-purpose microprocessors used in desktop computers and servers. Even with the scaling down of the supply voltage, power dissipation has

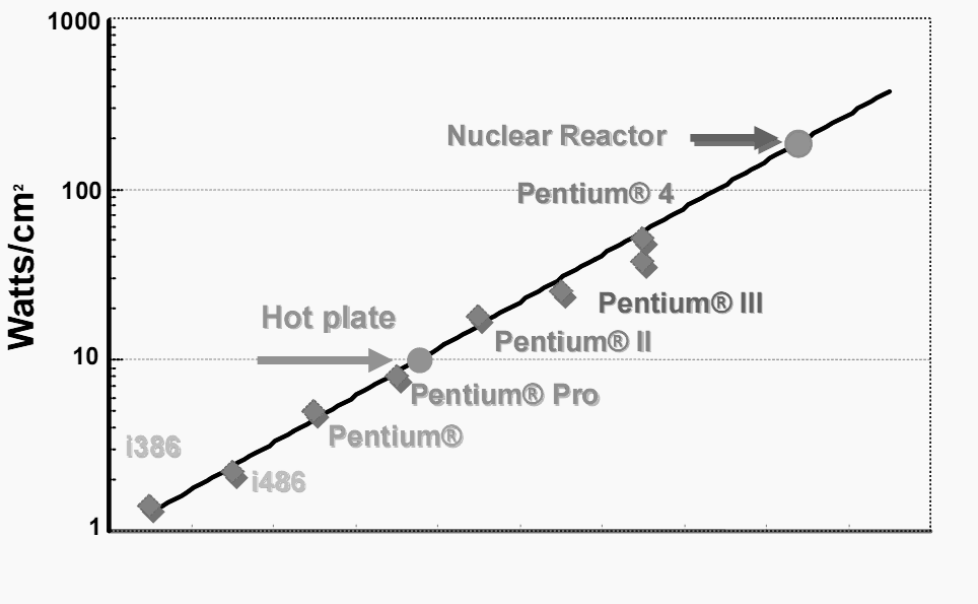

Figure 1.1: Power density for the Intel-32 family

not come down. Figure 1.1 shows the power density for several commercial processors. As it is shown in the figure, the trend is to increase the power density to levels where the cooling mechanisms are unlikely to be effective enough.

2. Portable battery-powered applications of the past were characterized by low computational requirements. The last few years have seen the emergence of portable applications that require a greater amount of processing power. Two vanguards of this processing model are the notebook computer and the personal digital assistant (PDA).

As a result, today, it is widely accepted that power efficiency is a design goal at par in importance with miniaturization and performance. In spite of this acceptance, the practice of low-power methodologies is being adopted at a slow pace due to the widespread changes

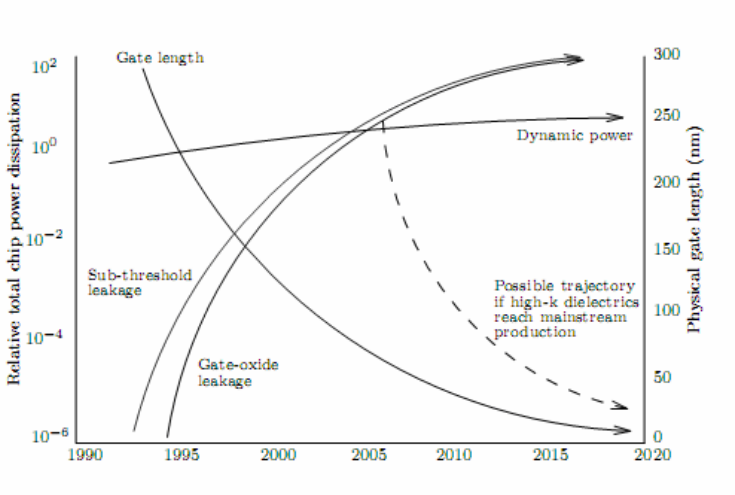

Figure 1.2: Projected development of power consumption over technology generations [44]

called for by these methodologies. Minimizing energy consumption and power dissipation calls for conscious effort at each abstraction level and at each phase of the design process.

While technology scaling has made it possible to put more transistors onto a single chip, at the same time allowing them to run faster, new complications continue to arise [1]. The supply voltage, being one of the critical parameters, has been reduced according to the characteristics of the shrinking MOS device. Therefore, to maintain the transistor switching speed, the threshold voltage is also scaled down at the same rate as the supply voltage. As a result, leakage currents increase dramatically with each technology generation. Some researchers predict a 7.5-fold increase in the leakage current and a 5-fold increase in total energy dissipation for every new microprocessor chip generation [64]. As the leakage current increases faster, it will become the major component in the total power dissipation. Figure 1.2 shows that static power consumption reaching the level of dynamic power consumption within a few years time. Leakage power is becoming a serious problem that needs handling at all levels of abstraction.

1.1 Problem Statement

In the presence of non-negligible leakage power, the way to design architectures for low power consumption may have changed. This master's thesis represents one step towards exploring low power design again. It concentrates on trying to reduce the leakage power at the architectural level.

1.2 Contribution of Research

We have developed a new Hardware-Software Architecture for the processors that helps in leakage power reduction for future higher-leakage technologies by using pipeline stalls. The architecture includes a power block that inserts NOPs in the processor in order to reduce the frequency of operation. Simultaneously when the NOP is inserted the different blocks of the processor are put into low-power mode, such as sleep mode or drowsy mode, by power-gating technique given in the literature.

1.3 Organization of the thesis

The thesis is organised as follows. In chapter 2, we survey the different power reduction techniques that has been used at different abstraction levels such as technology level, layout level, circuit level, gate level, architecture level and algorithm level. Chapter 3 explains the new hardware-software technique for leakage power reduction. The modified low-power architecture of the processor and the results are discussed in chapter 4. The thesis is concluded with the insight on the future work in chapter 5.

Chapter 2

Background

This chapter describes low-power design techniques at abstraction levels ranging from layout and technology to architecture and system. By reading this chapter, it should become clear that high-level design decisions - those made at the architecture or system level - have the most dramatic impact on power consumption.

2.1 SOURCES OF POWER CONSUMPTION

Power dissipated in CMOS circuits consists of several components as indicated below [15] [112]:

$$P_{total} = P_{switching} + P_{shortcircuit} + P_{static} + P_{leakage} \tag{2.1}$$

The individual components represent the power required to charge or switch a capacitive load ($P_{switching}$), short circuit power consumed during output transitions of a CMOS gate as the input switches ($P_{shortcircuit}$), static power consumed by the input switches (P_{static}), and leakage power consumed by the device ($P_{leakage}$). Components $P_{switching}$ and $P_{shortcircuit}$ are present when a device is actively changing state, while the components P_{static} and $P_{leakage}$ are present regardless of state changes.

The largest active component is $P_{switching}$. It is caused due to the logic transitions. As the transistors in the digital CMOS circuit transition back and forth between the two logic levels, the parasitic capacitances are charged and discharged. Current flows through the channel resistance of the transistors, and electrical energy is converted into heat and dissipated away (Fig. 2.1)

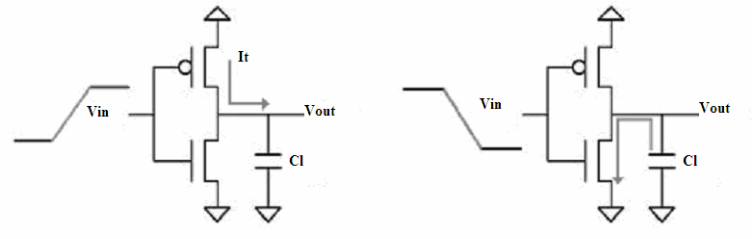

Figure 2.1: Circuit transition currents (left: charge, right: discharge)

$P_{switching}$ is defined as,

$$P_{switching} = C * V_{dd} * V_{swing} * \alpha * f \tag{2.2}$$

Where C represents the capacitance being switched, V_{dd} is the supply voltage, V_{swing} corresponds to the change in the voltage level of the switched capacitance, α represents a switching activity factor based on the probability of an output transition and, f represents the frequency of operation. The product $\alpha * C$ is also referred to as the effective switched capacitance, or C_{eff}. In most circuits, V_{swing} is equal to V_{dd}, so (2.2) is commonly written as

$$P_{switching} = C_{eff} * V_{dd}^2 * f \tag{2.3}$$

The term $P_{shortcircuit}$ occurs when short-circuit currents flow directly from supply to ground when both n-subnetwork and p-subnetwork of a CMOS gate conduct simultaneously (Fig.2.2). With the input(s) to the gate stable at either logic level, only one of the two subnetworks conduct and no short-circuit currents flow. But when the output of the gate is changing in response to the change in the input, both subnetworks conduct simultaneously for a brief interval. The duration of the interval and the short circuit dissipation both depend on the input and the output transition (rise or fall) times.

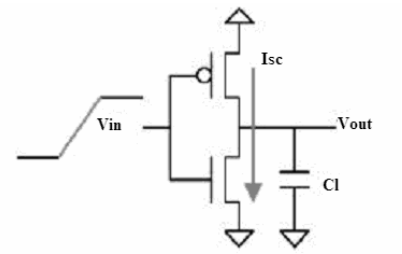

Figure 2.2: Short circuit currents

$P_{shortcircuit}$ has a complicated derivation, but in a simplified form can be written as [16],

$$P_{shortcircuit} = I_{mean} * V_{dd} \tag{2.4}$$

Where I_{mean} represents the average current drawn during the input transition. I_{mean} is minimized for a single gate with short input rise and fall times, and with long output transition times, thus presenting a trade off in device sizing. When a set of gates is considered, it is generally optimal to target equal input and output transition times. For large devices such as input-output (I/O) buffers or clock drivers, special design considerations are often used to minimize the overlap current [109]. For properly sized and ratioed gates, the contribution to overall dynamic power due to $P_{shortcircuit}$ is on the order of 10 % -20 %, although this factor may increase with increased device scaling [110].

P_{static} is not usually a factor in pure CMOS designs, since static current is not drawn by a CMOS gate, but certain circuit structures such as sense amplifiers, voltage references, and constant current sources do exist in CMOS systems and contribute to overall power.

$P_{leakage}$ is due to leakage currents from reversed biased PN junctions associated with the source and drain of MOS transistors, as well as subthreshold conduction currents. The leakage current flows when the input(s) to, and therefore the outputs of, a gate are not

changing (Fig. 2.3). The leakage component is proportional to device area and temperature. The subthreshold leakage component is strongly dependent on device threshold voltages, and becomes an important factor as power supply voltage scaling is used to lower power. For systems with a high ratio of standby operation to active operation, $P_{leakage}$ may be the dominant factor in determining overall battery life as lower the threshold voltage, the lower the degree to which MOSFETs in the logic gates are turned off and the higher is the standby leakage current.

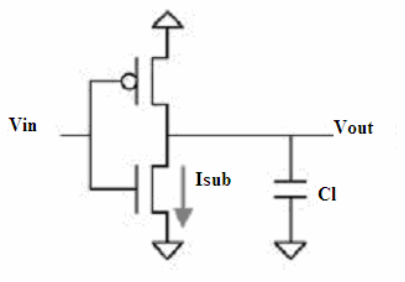

Figure 2.3: Leakage Currents

Minimization of these components of power dissipation is important in designing low-power systems, and there are complex interactions that require trade offs to be made involving each.

2.2 DEGREES OF FREEDOM

Active power minimization involves reducing the magnitude of each of the components in equation (2.2): voltage, physical capacitance, and activity. Optimizing for power invariably involves an attempt to reduce one or more of these factors. Unfortunately, these parameters cannot be optimized independently as they are not completely orthogonal. This

section briefly discusses each of the factors, describing their relative importance, as well as the interactions that complicate the power optimization process [22].

2.2.1 Voltage

With its quadratic relationship to power, voltage reduction offers the most direct and dramatic means of minimizing energy consumption. Without requiring any special circuits or technologies, a factor of two reduction in supply voltage yields a factor of four decrease in energy (see Figure 2.4(a) [18]). Furthermore, this power reduction is a global effect, experienced not only in one sub-circuit or block of the chip, but throughout the entire design. Because of this quadratic relationship, designers of low-power systems are often willing to sacrifice increased physical capacitance or circuit activity for reduced voltage. Unfortunately, supply voltage cannot be decreased without bound. In fact, several factors other than power influence selection of a system supply voltage. The primary factors are performance requirements and compatibility issues. As supply voltage is lowered, circuit

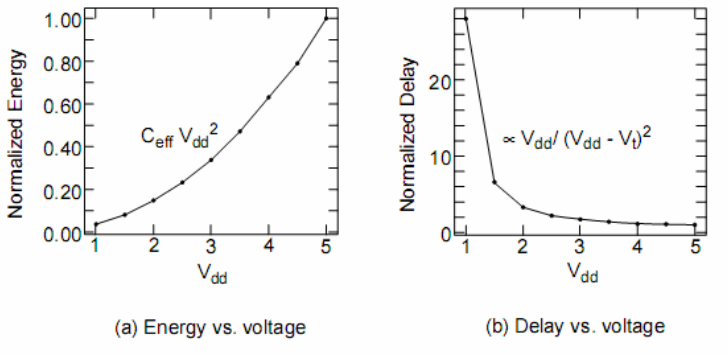

Figure 2.4: Energy and Delay as a function of supply voltage

delays increase (see Figure 2.4(b)) leading to reduced system performance. To the first order, device currents are given by:

$$I_{dd} = \frac{\mu C_{ox}}{2} \frac{W}{L} (V_{dd} - V_t)^2 \tag{2.5}$$

This leads to circuit delays of the order:

$$t = \frac{CV_{dd}}{I_{dd}} \propto \frac{V_{dd}}{(V_{dd} - V_t)^2} \tag{2.6}$$

So, for $V_{dd} >> V_t$ delays increase linearly with decreasing voltage. In order to meet system performance requirements, these delay increases cannot go unchecked. Some techniques must be applied, either technological or architectural, in order to compensate for this effect. As V_{dd} approaches the threshold voltage, however, delay penalties simply become unmanageable, limiting the advantages of going below a supply voltage of about $2V_t$.

Performance is not, however, the only limiting criterion. When going to non-standard voltage supplies, there is also the issue of compatibility and inter-operability. Unless the entire system is being designed completely from scratch it is likely that some amount of communications will be required with components operating at a standard voltage. This dilemma is lessened by the availability of highly efficient ($> 90\%$) DC-DC level converters, but still there is some cost involved in supporting several different supply voltages [98]. This suggests that it might be advantageous for designers to support only a small number of distinct intrasystem voltages. For example, custom chips in the system could be designed to operate off a single low voltage (e.g., $2V_t$) with level shifting only required for communication with the outside world. To account for parameter variations within and between chips, the supply would need to be set relative to the worst-case threshold, $V_{t,max}$.

To summarize, reducing supply voltage is paramount to lowering power consumption, and it often makes sense to increase physical capacitance and circuit activity in order to further reduce voltage. There are, however, limiting factors such as minimum performance and compatibility requirements that limit voltage scaling. These factors will likely lead designers to fix the voltage within a system. Once the supply has been fixed, it remains to address the issues of minimizing physical capacitance and activity at that operating voltage. The next two sections address these topics.

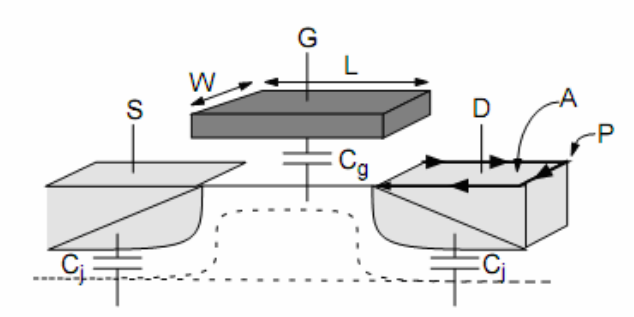

Figure 2.5: Primary sources of device capacitance

2.2.2 Physical Capacitance

Dynamic power consumption depends linearly on the physical capacitance being switched. So, in addition to operating at low voltages, minimizing capacitance offers another technique for reducing power consumption. In order to properly evaluate this opportunity we must first understand what factors contribute to the physical capacitance of a circuit. Then we can consider how those factors can be manipulated to reduce power. The physical capacitance in CMOS circuits stems from two primary sources: devices and interconnect. For devices, the most significant contributions come from the gate and junction capacitances as shown in Figure 2.5. The capacitance associated with the thin gate oxide of the transistor

is usually the larger of the two. This term can be approximated as a parallel-plate (area) capacitance between the gate and the substrate or channel:

$$c_g = WLC_{ox} = WL\frac{\epsilon_{ox}}{t_{ox}} \tag{2.7}$$

In addition, source/drain junction capacitances contribute to the overall device capacitance. These capacitances have both an area and a perimeter component and are non-linear with the voltage across the junction:

$$C_j(V) = AC_{j0}(1 - \frac{V}{\phi_0})^{-m} + PC_{jsw0}(1 - \frac{V}{\phi_0})^{-m} \tag{2.8}$$

Where A and P are the source/drain area and perimeter, C_{j0} and C_{jsw0} are equilibrium bottomwall and sidewall capacitances, ϕ_0 is the junction barrier potential,and m is the junction grading coefficient.

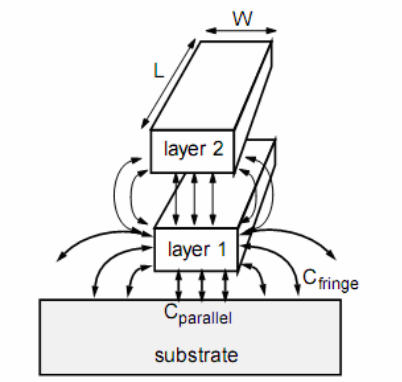

Figure 2.6: Sources of interconnect capacitance

Often, this non-linear capacitance is approximated by a large-signal equivalent linearized capacitance given by:

$$C_{jeq} = \frac{\int_{V_0}^{V_1} C_j(V)dV}{V_1 - V_0} \tag{2.9}$$

Where V_0 and V_1 describe the range of typical operating voltages for the junction. In past technologies, device capacitances dominated over interconnect parasitics. As technologies continue to scale down, however, this no longer holds true and we must consider the contribution of interconnect to the overall physical capacitance. For the interconnect, there is the capacitance between each metalization layer and the substrate, as well as coupling capacitances between the layers themselves (see Figure 2.6). Each of these capacitances in turn has two components: a parallel-plate component and a fringing component:

$$C_w = WLC_p + 2(W + L)C_f \tag{2.10}$$

Historically, the parallel-plate component, which increases linearly with both the width and the length of the wire, has been dominant. The fringing component starts to become significant, however, as the interconnect width becomes narrower and narrower relative to the wire thickness [13]. With this understanding, we can now consider how to reduce physical capacitance. From the previous discussion, we recognize that capacitances can be kept at a minimum by using small devices and short wires. As with voltage, however, we are not free to optimize capacitance independently. For example, reducing device sizes will not only reduce physical capacitance, but will also reduce the current drive of the transistors, making the circuit operate more slowly. This loss in performance might prevent us from lowering V_{dd} as much as we might otherwise be able to do. In this scenario, we are giving up a possible quadratic reduction in power through voltage scaling for a linear reduction

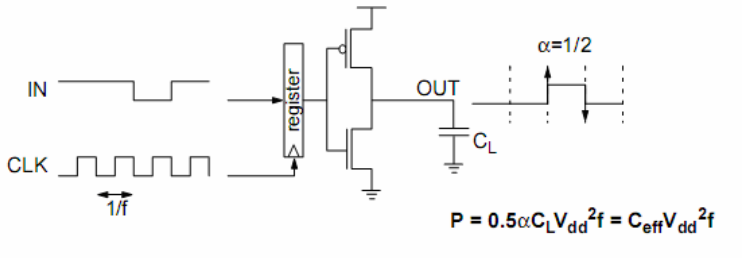

Figure 2.7: Interpretation of switching activity in synchronous systems

through capacitance scaling. So, if the designer is free to scale voltage it does not make sense to minimize physical capacitance without considering the side effects. Similar arguments can be applied to interconnect capacitance. If voltage and/or activity can be significantly reduced by allowing some increase in physical interconnect capacitance, then this may result in a net decrease in power. The key point to recognize is that low-power design is a joint optimization process in which the variables cannot be manipulated independently.

2.2.3 Activity

In addition to voltage and physical capacitance, switching activity also influences dynamic power consumption. A chip can contain a huge amount of physical capacitance, but if it does not switch then no dynamic power will be consumed. The activity determines how often this switching occurs. As mentioned above, there are two components to switching activity. The first is the data rate, f, which reflects how often, on average, new data arrives at each node. This data might or might not be different from the previous data value. In this sense, the data rate f describes how often on average switching could occur. For example, in synchronous systems f might correspond to the clock frequency (see Figure 2.7).

The second component of activity is the data activity, α. This factor corresponds to the expected number of transitions that will be triggered by the arrival of each new piece of data. So, while f determines the average periodicity of data arrivals, α determines how many transitions each arrival will spark. For circuits that don't experience glitching, α can be interpreted as the probability that a transition will occur during a single data period. For certain logic styles, however, glitching can be an important source of signal activity and, therefore, deserves some mention here [17]. Glitching refers to spurious and unwanted transitions that occur before a node settles down to its final, steady-state value. Glitching often arises when paths with unbalanced propagation delays converge at the same point in the circuit. Since glitching can cause a node to make several power consuming transitions instead of one (i.e. $\alpha > 1$) it should be avoided whenever possible. The data activity α can be combined with the physical capacitance C to obtain an effective capacitance, C_{eff} $= \alpha\ C/2$, which describes the average capacitance charged during each $1/f$ data period. This reflects the fact that neither the physical capacitance nor the activity alone determines dynamic power consumption. Instead, it is the effective capacitance, which combines the two, that truly determines the power consumed by a CMOS circuit:

$$P = \frac{1}{2}\alpha C {V_{dd}}^2 f = C_{eff} {V_{dd}}^2 f \tag{2.11}$$

This discussion provides the first indication that data statistics can have a significant effect on power consumption. As with voltage and physical capacitance, we can consider techniques for reducing switching activity as a means of saving power. For example, certain data representations, such as sign-magnitude, have an inherently lower activity than

two's-complement [27]. Since sign-magnitude arithmetic is much more complex than two's-complement, however, there is a price to be paid for the reduced activity in terms of higher physical capacitance. This is yet another indication that low-power design is truly a joint optimization problem. In particular, optimization of activity cannot be undertaken independently without consideration for the impact on voltage and capacitance.

2.3 RECURRING THEMES IN LOW POWER DESIGN

Sections 2.1 and 2.2 have provided a strong foundation from which to consider low-power CMOS design. Specifically, Section 2.1 derived the classical expression for dynamic power consumption in CMOS. This led to the realization that three primary parameters: voltage, physical capacitance, and activity determine the average power consumption of a digital CMOS circuit. Section 2.2 then went on to describe each of these factors individually, while emphasizing that design for low-power must involve a joint rather than independent optimization of these three parameters. The upcoming sections present specific power reduction techniques applicable at various levels of abstraction. Many of these techniques follow a small number of common themes. The three principle themes are trading area/performance for power, avoiding waste, and exploiting locality. Probably the most important theme is trading area/performance for power. As mentioned in Section 2.2.1, power can be reduced by decreasing the system supply voltage and allowing the performance of the system to degrade. This is an example of trading performance for power. If the system designer is not willing to give up the performance, he can consider applying techniques such as parallel processing to maintain performance at low voltage. Since many of these techniques incur an area penalty, we can think of this as trading area for

power. Another recurring low-power theme involves avoiding waste. For example, clocking modules when they are idle is a waste of power. Glitching is another example of wasted power and can be avoided by path balancing and choice of logic family. Other strategies for avoiding waste include using dedicated rather than programmable hardware and reducing control overhead by using regular algorithms and architectures. Avoiding waste can also take the form of designing systems to meet, rather than beat, performance requirements. If an application requires 25 MIPS of processing performance, there is no advantage gained by implementing a 50 MIPS processor at twice the power. Exploiting locality is another important theme of low-power design. Global operations inherently consume a lot of power. Data must be transferred from one part of the chip to another at the expense of switching large bus capacitances. Furthermore, in poorly partitioned designs the same data might need to be stored in many parts of the chip, wasting still more power. In contrast, a design partitioned to exploit locality of reference can minimize the amount of expensive global communications employed in favor of much less costly local interconnect networks. While not all low-power techniques can be classified as trading-off area/performance for power, avoiding waste, and exploiting locality these basic themes do describe many of the strategies that will be presented in the remainder of this chapter. The organization of these upcoming sections is by level of abstraction. Specifically, beginning with Section 2.4 and ending with Section 2.9, they cover low-power design methodologies for the technology, layout, circuit, gate, architecture, and algorithm levels, respectively.

2.4 TECHNOLOGY LEVEL

At the lowest level of abstraction we can consider low-power design strategies in the context of both packaging and process technologies.

2.4.1 Packaging

Often a significant fraction of the total chip power consumption can be attributed not to core processing but to driving large off-chip capacitances. This is not surprising since off-chip capacitances are on the order of tens of picofarads while on-chip capacitances are in the tens of femtofarads. For conventional packaging technologies, Bakoglu suggests that pins contribute approximately 13-14 pF of capacitance each (10 pF for the pad and 3-4 pF for the printed circuit board traces) [13]. Since dynamic power is proportional to capacitance, I/O power can be a significant portion of overall chip power consumption. The notion that I/O capacitance at the chip level can account for as much as 1/4 to 1/2 of the overall system power dissipation suggests that reduction of I/O power is a high priority in multi-chip systems. If the large capacitances associated with inter-chip I/O were drastically reduced, the I/O component of system power consumption would be reduced proportionally. Packaging technology can have a dramatic impact on the physical capacitance involved in off-chip communications. For example, multi-chip modules or MCM's offer a drastic reduction in the physical capacitance of off-chip wiring. In an MCM, all of the chips comprising the system are mounted on a single substrate, and the entire module is placed in a single package. Utilizing this technology, inter-chip I/O capacitances are reduced to the same order as on-chip capacitances [20]. This is due not only to the elimination of the highly capacitive PCB traces, but also to the minimization of on-chip pad driver capacitances due

to reduced off-chip load driving requirements. Thus, utilizing MCM technology, the I/O component of system power consumption can be kept at a minimum, shifting the focus of power optimization from I/O considerations to chip core considerations [23]. Actually, low-power operation is only one of the advantages of MCM technology. In addition, MCM's with their reduced chip-level interconnect lengths and capacitances can significantly reduce system delays resulting in higher performance, which can then be traded for lower power at the designer's discretion [13]. So, selection of a packaging technology can have an important effect on system power consumption.

2.4.2 Process

In addition to packaging considerations, process (or fabrication) technology plays an important role in determining power consumption. This section presents two important process-based techniques for reducing power consumption: technology scaling and threshold voltage scaling.

TECHNOLOGY SCALING

Scaling of physical dimensions is a well-known technique for reducing circuit power consumption. Basically, scaling involves reducing all vertical and horizontal dimensions by a factor, S, greater than one. Thus, transistor widths and lengths are reduced, oxide thicknesses are reduced, depletion region widths are reduced, interconnect widths and thicknesses are reduced, etc. The first-order effects of scaling can be fairly easily derived [12, 35]. Device gate capacitances are of the form $C_g = W\ L\ C_{ox}$. If we scale down W, L, and t_{ox} by S, then this capacitance will scale down by S as well. Consequently, if system data rates

and supply voltages remain unchanged, this factor of S reduction in capacitance is passed on directly to power:

$$Fixed\ Performance, Fixed\ Voltage : P \propto \frac{1}{S} \tag{2.12}$$

To give a concrete example, at the 1994 International Solid-State Circuits Conference, MIPS Technologies attributed a 25% reduction in power consumption for their new 64b RISC processor solely to a migration from a 0.8μm to a 0.64μm technology [114]. The effect of scaling on delays is equally promising. Based on (eq 2.5), the transistor current drive increases linearly with S. As a result, propagation delays, which are proportional to capacitance and inversely proportional to drive current, scale down by a factor of S^2. Assuming we are only trying to maintain system throughput rather than increase it, the improvement in circuit performance can be traded for lower power by reducing the supply voltage. In particular, neglecting V_t effects, the voltage can be reduced by a factor of S^2. This results in a S^4 reduction in device currents, and along with the capacitance scaling leads to an S^5 reduction in power:

$$Fixed\ Performance, Variable\ Voltage : P \propto \frac{1}{S^5} \tag{2.13}$$

This discussion, however, ignores many important second-order effects. For example, as scaling continues, interconnect parasitics eventually begin to dominate and change the picture substantially. The resistance of a wire is proportional to its length and inversely proportional to its thickness and width. Since in this discussion we are considering the impact of technology scaling on a *fixed design*, the local and global wire lengths should

scale down by S along with the width and thickness of the wire. This means that the wire resistance should scale up by a factor of S overall. The wire capacitance is proportional to its width and length and inversely proportional to the oxide thickness. Consequently, the wire capacitance scales down by a factor of S. To summarize:

$$R_w \propto S \; and \; C_w \propto \frac{1}{S} \tag{2.14}$$

$$t_{wire} \propto RwCw \propto 1 \tag{2.15}$$

This means that, unlike gate delays, the intrinsic interconnect delay does not scale down with physical dimensions. So at some point interconnect delays will start to dominate over gate delays and it will no longer be possible to scale down the supply voltage. This means that once again power is reduced solely due to capacitance scaling:

$$Parasitics \; dominated : P \propto \frac{1}{S} \tag{2.16}$$

Actually, the situation is even worse since the above analysis did not consider second-order effects such as the fringing component of wire capacitance, which may actually grow with reduced dimensions. As a result, realistically speaking, power may not scale down at all, but instead may stay approximately constant with technology scaling or even increase:

$$Including \; 2nd - order \; effects : P \propto 1 \; or \; more \tag{2.17}$$

The conclusion is that technology scaling offers significant benefits in terms of power only up to a point. Once parasitics begin to dominate, the power improvements slack off or disappear completely. So we cannot rely on technology scaling to reduce power indefinitely. We must turn to other techniques for lowering power consumption.

THRESHOLD VOLTAGE REDUCTION

Many process parameters, aside from lithographic dimensions, can have a large impact on circuit performance. For example, at low supply voltages the value of the threshold voltage (V_t) is extremely important. Section 2.2.1 revealed that threshold voltage places a limit on the minimum supply voltage that can be used without incurring unreasonable delay penalties. Based on this, it would seem reasonable to consider reducing threshold voltages in a low-power process. Unfortunately, subthreshold conduction and noise margin considerations limit how low V_t can be set. Although devices are ideally "off" for gate voltages below V_t, in reality there is always some subthreshold conduction even for $V_{gs} < V_t$. The question is especially important for dynamic circuits for which subthreshold currents could causes erroneous charging or discharging of dynamic nodes. The relationship between gate voltage and subthreshold current is exponential. Each 0.1V reduction in V_{gs} below V_t reduces the subthreshold current by approximately one order of magnitude [74]. Therefore, in order to prevent static currents from dominating chip power and to ensure functionality of dynamic circuits, threshold voltages should be limited to a minimum of 0.3-0.5V. Unfortunately, dimensional and threshold scaling are not always viable options. Aside from the drawbacks of interconnect non-scalability, submicron effects, and subthreshold conduction, chip designers often don't have complete freedom to arbitrarily scale their fabrication technology. Instead, economic factors as well as the capabilities of their fabrication facilities

impose limits on minimum lithographic dimensions. For this reason, in order to achieve widespread acceptance, an ideal low-power methodology should not rely solely on technology scaling or specialized processing techniques. The methodology should be applicable not only to different technologies, but also to different circuit and logic styles. Whenever possible, scaling and circuit techniques should be combined with the high-level methodology to further reduce power consumption; however, the general low-power strategy should not require these tricks.

2.5 LAYOUT LEVEL

There are a number of layout-level techniques that can be applied to reduce power. The simplest of these techniques is to select upper level metals to route high activity signals. The higher level metals are physically separated from the substrate by a greater thickness of silicon dioxide. Since the physical capacitance of these wires decreases linearly with increasing t_{ox}, there is some advantage to routing the highest activity signals in the higher level metals. For example, in a typical process metal three will have about a 30% lower capacitance per unit area than metal two [84]. The DEC Alpha chip takes advantage of this fact by routing the high activity clock network primarily in third level metal [36]. It should be noted, however, that the technique is most effective for global rather than local routing, since connecting to a higher level metal requires more vias, which add area and capacitance to the circuit. Still, the concept of associating high activity signals with low physical capacitance nodes is an important one and appears in many different contexts in low-power design. For example, we can combine this notion with the locality theme of Section 2.3 to arrive at a general strategy for low-power placement and routing. The placement and

routing problem crops up in many different guises in VLSI design. Place and route can be performed on pads, functional blocks, standard cells, gate arrays, etc. Traditional placement involves minimizing area and delay. Minimizing delay, in turn, translates to minimizing the physical capacitance (or length) of wires. In contrast, placement for power concentrates on minimizing the activity-capacitance product rather than the capacitance alone. In summary, high activity wires should be kept short or, in a manner of speaking, local. Tools have been developed that use this basic strategy to achieve about an 18% reduction in power [31, 108].

2.6 CIRCUIT LEVEL

Many circuit techniques can lead to reduced power consumption. In this section, we go beyond the traditional synchronous fully-complementary static CMOS circuit style to consider the relative advantages and disadvantages of other design strategies. This section will consider topics related to low-power circuit design.

2.6.1 Dynamic Logic

In static logic, node voltages are always maintained by a conducting path from the node to one of the supply rails. In contrast, dynamic logic nodes go through periods during which there is no path to the rails, and voltages are maintained as charge dynamically stored on nodal capacitances. Figure 2.8 shows an implementation of a complex boolean expression in both static and dynamic logic. In the dynamic case, the clock period is divided into a precharge and an evaluation phase. During precharge, the output is charged to V_{dd}. Then, during the next clock phase, the NMOS tree evaluates the logic function and discharges the

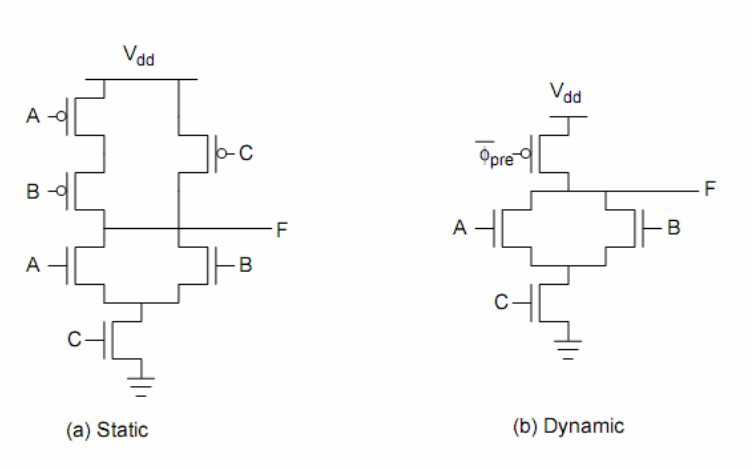

Figure 2.8: Static and Dynamic implementations of F = $\overline{(A+B)C}$

output node if necessary. Relative to static CMOS, dynamic logic has both advantages and disadvantages in terms of power.

Dynamic design styles have reduced device counts, do not experience short-circuit power dissipation, and are guaranteed to have a maximum of one transition per clock cycle unlike static gates which experience glitching, However, each of the precharge transistors in the chip must be driven by a clock signal requiring a dense clock distribution network and its associated capacitance and driving circuitry, which contributes significant power consumption to the chip. Also as each gate is influenced by the clock the issues of skew become important and difficult to handle.

2.6.2 Pass-Transistor Logic

As with dynamic logic, pass-transistor logic offers the possibility of reduced transistor counts. Figure 2.9 illustrates this fact with an equivalent pass-transistor implementation of the static logic function of Figure 2.8. Once again, the reduction in transistors results in

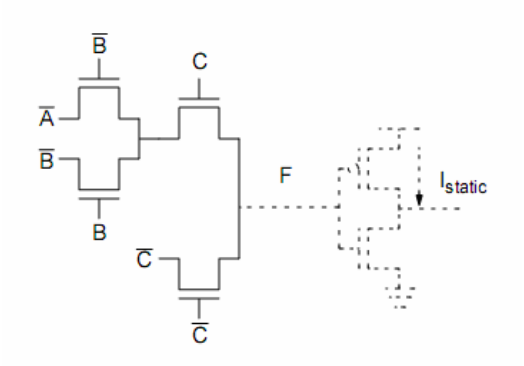

Figure 2.9: Complementary pass transistor implementation of F = $\overline{(A+B)C}$

lower capacitive loading from devices. This might make pass-transistor logic attractive as a low-power circuit style.

Like dynamic logic, however, pass-transistor circuits suffer from several drawbacks, at the voltages attractive for low-power design; the reduced current drive of pass-transistor logic networks becomes particularly troublesome. Low threshold processes can lessen this problem, but it is at the expense of robustness and static power dissipation.

2.6.3 Asynchronous Logic

Asynchronous logic refers to a circuit style employing no global clock signal for synchronization. Instead, synchronization is provided by handshaking circuitry used as an interface between gates (see Figure 2.10). While more common at the system level, asynchronous logic has failed to gain acceptance at the circuit level. This has been based on area and performance criteria. It is worthwhile to re-evaluate asynchronous circuits in the context of low power. The primary power advantages of asynchronous logic can be classified as avoiding waste. The clock signal in synchronous logic contains no information; therefore, power associated with the clock driver and distribution network is in some sense

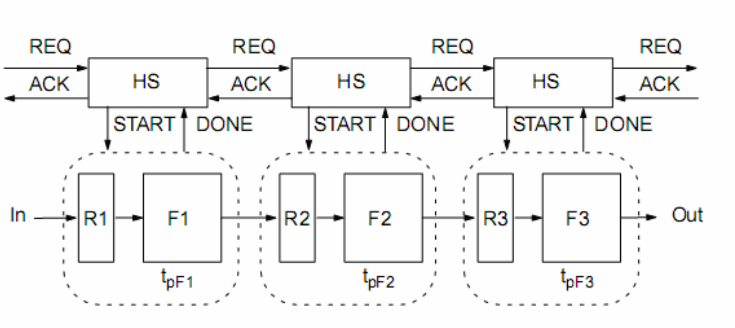

Figure 2.10: Asynchronous circuits with handshaking

wasted. Avoiding this power consumption component might offer significant benefits. In addition, asynchronous logic uses completion signals, thereby avoiding glitching, another form of wasted power. Finally, with no clock signal and with computation triggered by the presence of new data, asynchronous logic contains a sort of built in power-down mechanism for idle periods

At the small granularity with which it is commonly implemented, the overhead of the asynchronous interface circuitry dominates over the power saving attributes of the design style. It should be emphasized, however, that this is mainly a function of the granularity of the handshaking circuitry. It would certainly be worthwhile to consider using asynchronous techniques to eliminate the necessity of distributing a global clock between blocks of larger granularity.

2.6.4 Transistor Sizing

Regardless of the circuit style employed, the issue of transistor sizing for low power arises. The primary trade-off involved is between performance and cost - where cost is measured by area and power. Transistors with larger gate widths provide more current drive than smaller transistors. Unfortunately, they also contribute more device capacitance

to the circuit and, consequently, result in higher power dissipation. Moreover, larger devices experience more severe short-circuit currents, which should be avoided whenever possible. In addition, if all devices in a circuit are sized up, then the loading capacitance increases in the same proportion as the current drive, resulting in little performance improvement beyond the point of overcoming fixed parasitic capacitance components. In this sense, large transistors become self-loading and the benefit of large devices must be re-evaluated. A sensible low-power strategy is to use minimum size devices whenever possible. Along the critical path, however, devices should be sized up to overcome parasitics and meet performance requirements. Care should be taken in this sizing process to avoid the waste of self-loading [24]. By following this approach, Nagendra et al. found that the average power dissipation of a signed-digit adder could be reduced by 36% with a delay penalty of only 0.3% [76].

2.6.5 Design Style

Another decision which can have a large impact on the overall chip power consumption is the selection of a design style, e.g., full custom, gate array, standard cell, etc. Not surprisingly, full-custom design offers the best possibility of minimizing power consumption. In a custom design, all the principles of low-power including locality, regularity, and sizing can be applied optimally to individual circuits. Unfortunately, this is a costly alternative in terms of design time, and can rarely be employed exclusively as a design strategy. Other possible design styles include gate arrays and standard cells. Gate arrays offer one alternative for reducing design cycles at the expense of area, power, and performance. While not offering the flexibility of full-custom design, gate-array CAD tools could nevertheless be altered to

place increased emphasis on power. Standard cell synthesis is another commonly employed strategy for reducing design time. Current standard cell libraries and tools, however, offer little hope of achieving low power operation. In many ways, standard cells represent the antithesis of a low-power methodology. First and foremost, standard cells are often severely oversized. Most standard cell libraries were designed for maximum performance and worst-case loading from inter-cell routing. As a result, they experience significant self-loading and waste correspondingly significant amounts of power.

2.6.6 Dynamic Voltage Scaling

The other technique that is considered for power reduction at the circuit level is dynamic voltage scaling where a component is run at a less-than-maximum voltage in order to conserve power. Dynamic voltage scaling is most commonly used in laptops and other mobile devices, where energy comes from a battery and thus is limited. As explained in Section 2.2.1, the switching power dissipated by a chip decreases quadratically with voltage. A problem with this technique in embedded devices is that batteries have a certain voltage so extra circuitry must be introduced to scale the voltage down from the original value. This circuitry may take a while to settle at the desired voltage, hence adding a time-cost in the μs range on reducing the supply voltage. Dynamic frequency scaling is another power conservation technique that works on the same principles as dynamic voltage scaling. Both dynamic voltage scaling and dynamic frequency scaling can be used to prevent computer system overheating, which can result in program or operating system crashes, and possibly hardware damage. The speed at which a digital circuit can switch states - that is, to go

from "low" (VDD) to "high" (VSS) or vice versa - is proportional to the voltage differential in that circuit. Reducing the voltage means that circuits switch slower, reducing the maximum frequency at which that circuit can run. This, in turn, reduces the rate at which program instructions that can be issued, increasing program runtime.

2.6.7 Dual Threshold Voltage

It may be expected that dynamic voltage scaling will always reduce dynamic power dissipation in a long period of operating time since when the workload is reduced the supply voltage could also be reduced to save power. However, in deep submicron technologies, we need to begin to take leakage power consumptions into account as well, especially when design with dual threshold voltage cells are becoming widely adopted. A recent report [43] gives a comprehensive analysis of the consequences of applying Dynamic Voltage Scaling (DVS) to dual V_t cell design. A typical design scenario could be as follows. In the initial design, all cells used are low leakage (LL) cells to minimize power consumption. Then we could replace cells along critical paths by high-speed (HS) cells in order to shorten the delay. Thus we get a mixed cells design. And now DVS could be used to further decrease dynamic power dissipation when a larger delay could be tolerated. When applying DVS, we only get power gains if total power consumption in the mixed cell design is less than that in the original single LL cell design. According to the author M. Hans, subthreshold leakage is supply voltage dependent. An example [43] shows that after applying DVS the dynamic power dissipation is indeed decreased. However, the leakage consumption is increased at the same time. This tells us that DVS could have a negative impact on leakage consumption under certain circumstances and thus careful analysis needs to be done before making design

choices. The literature [10, 25, 33, 41, 52, 54, 70, 81, 95, 101, 107] gives more information on different experiments that has been tried using DVS, Transistor Sizing and Dual Threshold Voltage Techniques.

2.6.8 Dynamic Power Cutoff Technique (DPCT)

Dynamic Power Cutoff Technique (DPCT) is an active leakage power reduction technique. In this technique first the switching window for each gate, during which a gate makes its transitions, is identified by static timing analysis. Then, the circuit is optimally partitioned into different groups based on the minimal switching window (MSW) of each gate. Finally, power cut off transistors are inserted into each group to control the power connections of that group. Each group is turned on only long enough for a wavefront of changing signals to propagate through that group. Since each gate is only turned on during a small timing window within each clock cycle, this significantly reduces active leakage power. This technique can also save standby leakage and dynamic power. Results on ISCAS'85 benchmark circuits modeled using 70nm Berkeley Predictive Models show up to 90% active leakage, 99% standby leakage, 54% dynamic power, and 72% total power savings [115, 116]. However, this technique being new, its effect on the noise margin, power grid design and layout is still not known.

2.6.9 Retiming

Retiming is a classic logic optimization technique for synchronous circuits. Retiming is the technique of moving the structural location of latches or registers in a digital circuit to improve its performance, area, and/or power characteristics in such a way that preserves

its functional behavior at its outputs. When originally introduced [58, 59, 60], its emphasis was on the application to systolic systems. A subsequent paper [61] fully revisits the concept of retiming and shows how generic synchronous circuits can benefit from it under three main optimality criteria: (1) minimize the circuit clock period by adding/removing storage elements, (2) minimize the circuit area by reducing the number of storage elements, and (3) minimize circuit area under a maximum clock-period constraint. In the last two decades, Retiming has been adopted as a key optimization technique within every major logic synthesis tool both in academia and industry.

The other circuit level techniques have been discussed several authors [26, 30, 63, 100].

2.7 GATE LEVEL

As in the case of the circuit level, there are gate-level techniques that can be applied successfully to reduce power consumption. Once again these techniques reflect the themes of trading performance and area for power, avoiding waste, and exploiting locality. In this section we discuss a number of gate-level techniques and give some quantitative indication of their impact on power. In particular, this section presents techniques for technology mapping, glitching and activity reduction, input vector control technique and methods for exploiting concurrency and redundancy in the context of low-power design.

2.7.1 Technology Decomposition and Mapping

Technology decomposition and mapping refers to the process of transforming a gate-level boolean description of a logic network into a CMOS circuit. For a given gate-level

network there may be many possible circuit-level implementations. For instance, a three-input NAND can be implemented as a single complex CMOS gate or as a cascade of simpler two-input gates. Each mapping may result in different signal activities, as well as physical capacitances. For example, complex gates tend to exhibit an overall lower physical capacitance since more signals are confined to internal nodes rather than to the more heavily loaded output nodes. The concept of technology mapping for low-power is to first decompose the boolean network such that switching activity is minimized, and then to hide any high activity nodes inside complex CMOS gates. In this way, rapidly switching signals are mapped to the low capacitance internal nodes, thereby reducing power consumption. Making a gate too complex, however, can slow the circuit, resulting in a trade-off of performance for power. Several technology mapping algorithms for low power have been developed and offer an average power reduction of 12% [102] to 21% [104].

2.7.2 Activity Postponement

While technology mapping attempts to minimize the activity-capacitance product, other gate-level strategies focus on reducing activity alone. For example, an operation as simple as reordering the inputs to a boolean network can in some cases reduce the total network activity (see Figure 2.11) [56]. The basic concept is to postpone introduction of high activity signals as long as possible. In this way, the fewest gates are affected by the rapidly switching signals.

2.7.3 Glitch Reduction

Other gate-level activity reduction techniques focus on avoiding the wasted transitions associated with glitching. Figure 2.12 shows two implementations of the same logic function.

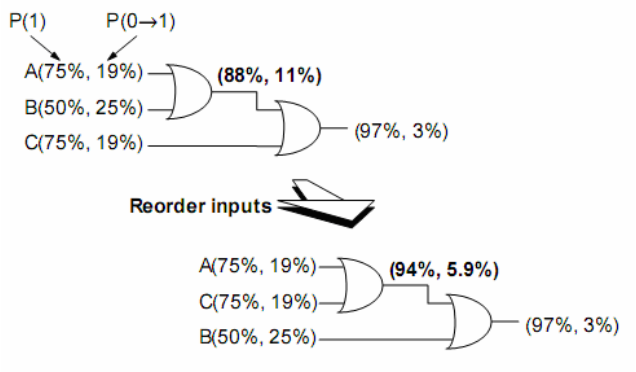

Figure 2.11: Input Reordering for activity reduction

One implementation employs a balanced tree structure, while the other uses a cascaded gate structure. If we assume equal input arrival times and gate delays, we find that the cascaded structure undergoes many more transitions than the tree structure before settling at its steady-state value. In particular, the arrival of the inputs may trigger a transition at the output of each of the gates. These output transitions may in turn trigger additional transitions for the gates within their fan-out. This reasoning leads to an upper-bound on glitching that is $O(N^2)$, where N is the depth of the logic network. In contrast, the path delays in the tree structure are all balanced, and therefore, each node makes a single transition and no power is wasted. This concept can be extended to derive optimum tree structures for the case of unequal arrival times as well [75]. Some studies have suggested that eliminating glitching in static circuits could reduce power consumption by as much as 15-20% [17].

Techniques for reducing glitch power have been described by several authors [5, 6, 49, 50, 66, 67, 68, 69, 86, 87, 88, 89, 90, 105, 106].

A mixed integer linear programming (MILP) technique [68] is used to minimize the leakage as well as glitch power consumption of a static CMOS circuit for any specified

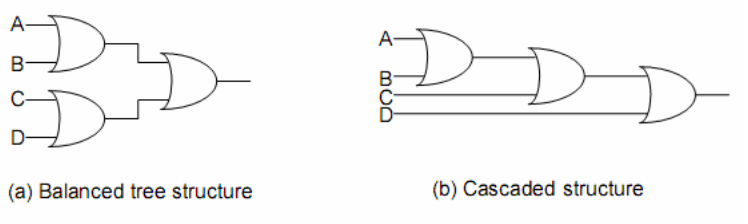

Figure 2.12: Cascaded versus Balanced tree gate structures

input to output delay. Using dual-threshold devices the number of high-threshold devices is maximized and a minimum number of delay elements are inserted to reduce the differential path delays below the inertial delays of incident gates. The key features of this method are that the constraint set size for the MILP model is linear in the circuit size and power-performance trade off is allowed. Experimental results for this technique shows 96%,40% and 70% reductions of leakage power, dynamic power and total power respectively for the benchmark circuit C7552 implemented in the 70nm BPTM CMOS technology.

In a CMOS circuit, energy consumption per signal transition at a node with capacitance C is $0.5CV^2$. Keeping the gate delays, internal to standard cells, fixed the authors of [106] determine the values of necessary routing delays to eliminate all glitches by either path delay balancing or inertial filtering. To implement these delays they insert the required amounts of resistances as customized feedthrough cells. In spite of the increased resistance in the circuit, the overall power is reduced because the resistive delays suppress glitches without increasing the $0.5CV^2$ power per transition, and no increase in the critical path delay is incurred. For the ISCAS '85 benchmark circuit, c2670, 30% saving has been achieved in average power consumption with 14% increase of the chip area.

Table 2.1: Leakage current values for different input combinations of a 3-input NAND gate

Input State	Subthreshold leakage (nA)	Gate Leakage (nA)	Total Leakage (nA)
000	0.49	6.58	7.07
001	0.81	19.68	21.49
010	0.81	6.79	7.60
011	2.68	34.78	37.46
100	0.81	3.15	3.96
101	2.68	16.8	19.48
110	2.68	1.84	4.52
111	16.85	45.3	62.15

2.7.4 Input Vector Control

Much research has been done to model and estimate the nominal leakage current of a circuit. Due to the stacking effect, the leakage of a circuit depends on its input combinations [3]. Table 2.1 shows the leakage components for all input combinations of a 3-input NAND gate.

Thus by finding a minimum leakage vector (MLV) and applying it to the circuit, we can guarantee the circuit turns into its low leakage state. Abdollahi et al. [3] have proposed a technique to identify the MLV and discussed several ways to apply the vector to the circuit. They first construct a Boolean network to compute the total leakage and then they use the SAT solver to find an input vector that results in the minimum leakage of the whole circuit. However, the effectiveness of the method does not rely on finding the MLV solely. Due to the limited access to internal nodes, it is very difficult to put the ideal value to every node especially for very complex circuits. Therefore the authors tried two ways to increase controllability of the internal nodes, namely adding multiplexers and modifying gates. Since both methods change the circuit to some extent, new Boolean networks need

to be constructed. Although the authors found an efficient way to identify the MLV and figured out ways to increase controllability, we could see that the actual application of input vector control is still limited by the primary inputs. And to put a sleep circuit back to normal will require extra memory elements to store the original states, thus incurring both area and delay penalty. Another controllability increasing method is given by Rahman and Chakrabarti [85].

2.7.5 Concurrency and Redundancy

The final technique discussed in this section is the use of concurrency and redundancy at the gate level in low-power design. The principal concept is to use concurrency and redundancy to improve performance and then to trade this performance for lower power by reducing the voltage supply. In some sense, path balancing, which was found to be useful for reducing glitching activity, can be thought of as a form of gate-level concurrent processing. Referring to Figure 2.12, the path balanced tree structure is characterized by several logic gates computing in parallel, with the gates in their fan-out combining the results. In contrast, for the linear, cascaded structure computations must take place sequentially since the results from the initial stages are needed to compute the output of the later stages. So by using concurrency, the tree structure achieves a shorter critical path than the cascaded structure - quantitatively, logarithmic as opposed to linear. This reduced critical path can be used to improve performance, or this performance can be traded for power by reducing the operating voltage until the delay of the logarithmic structure matches that of the linear structure.The majority of the techniques employing concurrency or redundancy incur an inherent penalty in area, as well as in physical capacitance and switching activity. At first

glance, a carry-select adder with 50 % more physical capacitance and activity than a ripple-carry adder might not seem low power at all. The key concept is to identify the design paradigm under which you are working: fixed voltage or variable voltage. If the voltage is allowed to vary, then it is typically worthwhile to sacrifice increased physical capacitance and activity for the quadratic power improvement offered by reduced voltage. If, however, the system voltage has been fixed, then there is nothing gained by employing a carry-select adder in place of a ripple-carry adder, unless the slower adder does not meet the timing constraints. So in such situation its better to use the least complex adder that meets the performance requirements. This falls under the category of avoiding waste [76].

2.8 ARCHITECTURE AND SYSTEM LEVELS

This chapter has repeatedly suggested that decisions made at a high level (architecture or system) will have a much larger impact on power than those made at a lower level (e.g., gate or circuit). This section provides some evidence to support this claim. In the terminology of this thesis, architecture refers to the register-transfer (RT) level of abstraction, where the primitives are blocks such as multipliers, adders, memories, and controllers. This level of abstraction is also referred to as the micro-architecture level. Having defined the terminology, this section discusses architectural or RT-level techniques for reducing power consumption.

2.8.1 Concurrent Processing

Perhaps the most important strategy for reducing power consumption involves employing concurrent processing at the architecture level. This is a direct trade-off of area and

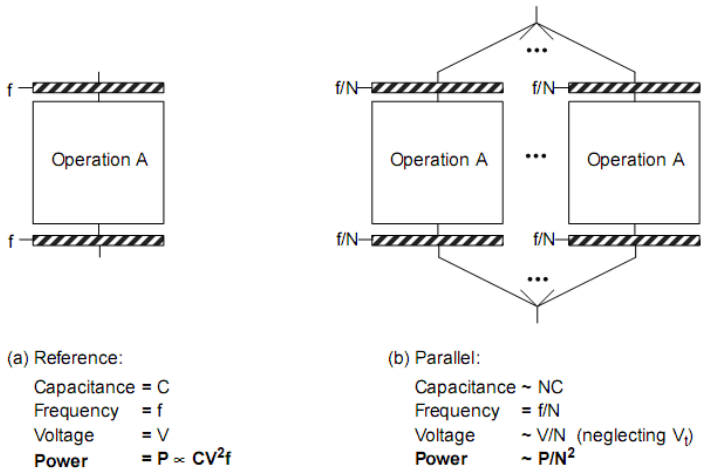

Figure 2.13: Voltage Scaling and Parallelism for low power

performance for power. In other words, the designer applies some well-known technique for improving performance such as parallelism or pipelining, and then swaps this higher performance for lower power by reducing the supply voltage.

PARALLELISM

As a quantitative example, consider the use of parallelism to perform some complex operation, A (see Figure 2.13(a)). The registers supplying operands and storing results for A are clocked at a frequency f. Further assume that algorithmic and data dependency constraints do not prevent concurrency in the calculations performed by A. When the computation of A is parallelized, Figure 2.13(b) results. The hardware comprising block A has been duplicated N times, resulting in N identical processors. Since there are now N processors, a throughput equal to that of sequential processor, A, can be maintained with a clocking frequency N times lower than that of A. That is, although each block will produce a result only $1/N_{th}$ as frequently as processor A, there are N such processors producing results. Consequently, identical throughput is maintained.

The key to this architecture's utility as a power saving configuration lies in this factor of N reduction in clocking frequency. In particular, with a clocking frequency of f/N, each individual processor can run N times slower. Since to the first order, delays vary roughly linearly with voltage supply, this corresponds to a possible factor of N reduction in supply voltage. Examining the power consumption relative to the single processor configuration, we see that capacitances have increased by a factor of N (due to hardware duplication), while frequency and supply voltage have been reduced by the same factor. Thus, since $P = CV^2f$, power consumption is reduced by the square of the concurrency factor, N:

$$N - way\ Concurrency : P \propto 1/(N)^2 \tag{2.18}$$

Hardware parallelism also has its disadvantages. For instance, complete hardware duplication entails a severe area penalty. In addition, there is hardware and interconnect overhead related to signal distribution at the processor inputs and signal merging at the outputs. These contribute to increased power consumption and tend to limit the utility of excessive parallelism. Also, the area requirements of full parallelism can be a limiting factor. Still, other forms of concurrent processing offer some of the power savings of parallelism at reduced cost.

PIPELINING

Pipelining is another form of concurrent computation that can be exploited for power reduction. An example of pipelining for low power is shown in Figure 2.14(b). In this situation, rather than duplicating hardware, concurrency is achieved by inserting pipeline registers, resulting in an N-stage pipelined version of processor A (assuming processor A can be pipelined to this extent). In this implementation, maintaining throughput requires

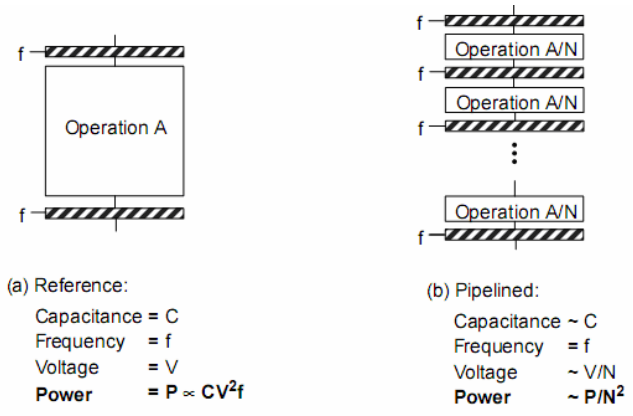

Figure 2.14: Voltage Scaling and Pipelining for low power

that we sustain clocking frequency, f. Ignoring the overhead of the pipeline registers, the capacitance, C, also remains constant. The advantage of this configuration is derived from the greatly reduced computational requirements between pipeline registers. Rather than performing the entire computation, A, within one clock cycle, only $1/N_{th}$ of A need be calculated per clock cycle. This allows a factor N reduction in supply voltage and,considering the constant C and f terms, the dynamic power consumption is reduced by N^2.

Thus, for both concurrency techniques - pipelining and parallelism - consumption is reduced by N, resulting in a first-order quadratic reduction in power consumption. As with parallelism, pipelining incurs some overhead, though not nearly as much. In a pipelined processor, for example, the pipeline registers represent an overhead in both power and area. First, each register must be clocked, adding to the capacitive loading of the clock network. As the pipeline depth increases, the area and capacitance associated with the pipeline registers approaches that of the actual processor stages. At that point, further pipelining becomes unattractive. Still, the overhead associated with pipelining is typically

much less than that of parallelism, making it a useful form of concurrent processing for power minimization.

A lot of other techniques have been combined with pipelining to reduce power in processors. Software constitutes a very important part of the system these days and a large portion of the functionality of the systems is in the form of instructions as opposed to gates so analyzing power consumption from the point of view of instructions is equally important.

Pipelined processors frequently insert NOP instructions into the pipe for generating delay or resolving dependency. NOP instruction is a No-Operation Instruction. It consumes only 1 clock cycle compared to other instructions and consumes the minimum power but the NOP energy significantly varies depending upon its position in the program.

[77] presents an approach for power virus generation using behavioral models for digital circuits. The technique presented converts the given behavioral model automatically to an integer (word-level) constraint model and employs an integer constraint solver to generate the required power virus vectors. Experiment results on the DLX processor showed two power virus sequences that consumed maximum power and the NOPs were predominant in these sequences which show that NOP energy significantly varies depending upon its position in the program as well as the instructions preceding and succeeding it.

P. Kamran et al. [65] describe a technique to optimize dynamic power consumption by eliminating useless transitions that are generated by the pipeline when a stall happens. For the NOP instruction to generate as few transitions as possible, the data part of the instruction is kept as the preceding instruction and in this way as a NOP instruction passes through a pipe, relative to the previous cycle, the same operations are performed on the

same data in all stages of the pipeline, therefore only a small number of transitions is generated as a result of the NOP insertion and propagation. By this technique the authors demonstrate up to 10% reduction in power consumption for some benchmarks at a cost of negligible performance and area overhead (below 0.1%).

2.8.2 Power Management

Any power consumed by a system that does not lead to useful results is wasted. For previous generations of chips, where power was not a major concern, this source of waste was typically ignored. Strategies for avoiding wasted power in systems are often called power management strategies. Some power management techniques available to designers include selective power down, sleep mode, and adaptive clocking/voltage schemes. Selective power down refers to deactivating processing modules that are not doing any useful work. A selective power down strategy requires additional logic to monitor the activity of the various modules within a system. Control lines signaling idle periods are then used to gate the clocks to the various system modules. As a result, idle modules are not clocked and no power is wasted. Some care must be taken in applying this strategy to dynamic circuits, for which clocking is used to maintain information stored on dynamic nodes. Two of the other methods that could be considered instead of complete power down are as follows:

CLOCK GATING:

Instead of switching off the power supply, the clock signal may be halted in idle devices. This reduces switching activity and therefore dynamic power consumption to zero. Inserting clock gates is not as great of an interference to the design as power supply shutdown and is applicable for applications where power shut down is no alternative. Clock gating won't

lessen power dissipation to zero since leakage power is unaffected. The designer has to take into account that the gate increases clock skew and makes testing more complicated. Lastly, it should be seen that glitches on the switch's control signal is prevented. For example, a glitch could cause a temporarily false clock turn off/on, which might add an extra rising edge to the clock signal behind the gate and as a result the behavior of the circuit is not preserved.

ENABLED FLIP-FLOPS:

As clock gating can be seen as a softer alternative to power supply shut down, enabled flip-flops are the next less aggressive (and less effective) strategy. Registers are replaced by a representative with an enable signal. By enabling these representatives, they behave like general registers. Disabled, the flip-flops' outputs are not changing, which reduces switching activity in the circuit. The most active signal, the clock, is still active though, ensuing a great deal of power dissipation. It can be said that power management based on enabled flip-flops can be beneficial, but an implementation based on gated-clocks is fundamentally superior.

LOW-POWER MODES:

Sleep mode is an extension of the selective power-down strategy. Here, the activity of the entire system is monitored rather than that of the individual modules. If the system has been idle for some predetermined time-out duration, then the entire system is shut down and enters what is known as sleep mode. During sleep mode the system inputs are monitored for activity, which will then trigger the system to wake up and resume processing. Since there is some overhead in time and power associated with entering and leaving sleep

mode, there are some trade-offs to be made in setting the length of the desired time-out period.

To reduce leakage power drowsy mode is another option under power-down strategy. When in drowsy mode, the information in the cache line is preserved; however, the line must be reinstated to a high-power mode before its contents can be accessed. A recent paper [40] shows that with simple architectural techniques, about 80%-90% of the cache lines can be maintained in a drowsy state without affecting performance by more than 1%.

At ISSCC'94, Intel, MIPS Technologies, and IBM all reported microprocessors that include selective power down and sleep-mode capabilities [14, 83, 94, 114]. IBM estimated that selective clocking saved 12-30 % in the power consumption of their 80MHz super-scalar PowerPC architecture [83]. In addition, Intel estimated that the combination of both techniques resulted in an overall 2x reduction in average power when running typical applications on their design [94].

Another power management strategy involves adapting clocking frequency and/or supply voltage to meet system performance requirements. Since the performance requirements of a system typically vary over time as the task it is performing changes, it is wasteful to run the system at maximum performance, even when a minimum of compute power is required. Adapting the clocking frequency or supply voltage of the system to reduce performance (and power) during these periods can result in substantial power savings. Since it is difficult to directly measure how much performance is actually required at a given point in time, some indirect performance feedback scheme must be devised. This can be in the form of clock slow-down instruction issued by the software application. MIPS Technologies

takes this approach in their 64b RISC processor, achieving a 4x reduction in power through reduced clock frequency [114].

To summarize, without careful management, large amounts of system power can be wasted by continuing computations during idle periods. A power-conscious system will avoid this source of waste either by powering down inactive modules or processors or by adapting the processing power of the system to meet, rather than exceed, the current requirements.

Many other and similar power saving techniques at the architectural level have been discussed by several authors [19, 32, 37, 47, 48, 57, 62, 71, 72, 73, 79, 91, 92, 96, 97, 99, 113].

2.8.3 Memory Partitioning

Farrahi et al. [39] propose a memory partitioning (also called segmentation) scheme that reduces power by exposing idleness in memory access. The functionality of memory is to store data when it is written and return it when read. Farrahi suggests to view memory not as a monolithic resource but as a collection of independent memory segments. Each segment has its own clock and refresh signals. Whenever a memory segment is idle, it can be put in a sleep mode where the clock is halted or no refreshes are transmitted. Memory is idle, when no useful information is stored in it. It should be kept in mind that memory is not idle, when it is not accessed. It might store vital information which would be lost when the memory is turned off. It might store unimportant information though. A lifetime can be assigned to each variable in a memory element. It defines a time interval which starts when a variable is written, and ends when the variable is last read. A segment is called idle, when it contains no live variables. The partitioning technique attempts to store

variables which have overlapping lifetimes in the same segment. Due to this approach, idle time of memory segments is increased and power dissipation is reduced. As with processing hardware, a distributed array of small local memories is often more power efficient than a shared, global memory subsystem. In particular, the energy consumed in accessing a memory is approximately proportional to the number of words stored in that memory. If this number can be reduced by partitioning the memory, then the total power associated with memory accesses will also be reduced. For example, assume 10 independent processors need to access one piece of data each. A single, shared memory will need to contain 10 words and each access will take 10 energy units for a total of 100 units of energy. On the other hand, if each processor utilizes a local memory containing the one piece of data it requires, then each of the 10 accesses will consume only one unit of energy for a total of 10 units - a factor of 10 savings in power. This example, though idealized, suggests the advantage of local or distributed memory structures over global or shared memories.

2.8.4 Programmability

The previous section suggested that distributed processors, which can be optimized for specific tasks, might consume less power than general-purpose, programmable processors that must execute many different tasks. This observation was made in the context of distributed versus centralized processing; however, it brings up the important issue of programmable versus dedicated hardware. As an example, consider the implementation of a linear, time-invariant filter. Such filters involve multiplication of variables by constants. All required constant multiplications could be implemented on a single array multiplier. This is a programmable scenario since the multiplier can be programmed to execute any of

the different constant multiplications on the same hardware. Alternatively, we can consider implementing each of the constant multiplications on dedicated hardware. In this case, since the multipliers each need to perform multiplication by a single, fixed coefficient, the array multipliers can be reduced to add-shift operations with an adder required only for the 1 bits in the coefficient and the shifting implemented by routing. For coefficients with relatively few 1's, this dedicated implementation can result in a greatly reduced power consumption, since the waste and overhead associated with the programmable array multiplier is avoided. This approach was taken by Chandrakasan for the implementation of a video color space converter for translating YIQ images to RGB [29]. The algorithm consists of a 3×3 constant matrix multiplication (i.e., nine multiplications by constant coefficients). Chandrakasan not only replaced the array multipliers with dedicated add-shift hardware, but also scaled the coefficients to minimize the number of 1 bits in the algorithm. The resulting chip consumed only 1.5 mW at 1.1V. As this example demonstrates, avoiding excessive or unnecessary programmability in hardware implementations can lead to significant reductions in power. Programmability does, however, offer some advantages. For instance, dedicated hardware typically imposes some area penalty. In the above case, the nine multiplications would each require their own unique hardware blocks. In contrast, in a programmable implementation a single array multiplier could be used to perform all nine multiplications. Another advantage of programmable hardware is that it simplifies the process of making design revisions and extensions. The behavior of a programmable component can be altered merely by changing the instructions issued by the software driving the device. A version of the chip relying primarily on dedicated hardware, however, might require extensive redesign efforts.

2.8.5 Data Representation

Another architectural decision that can impact power consumption is the choice of data representation. In making this decision, the designer typically has several different alternatives from which to choose, e.g., fixed-point vs. floating-point, sign-magnitude vs. two's-complement, and uncoded vs. encoded data. Each of these decisions involves a trade-off in accuracy, ease of design, performance, and power. This section discusses some of the issues involved in selecting a data representation for low power. The most obvious trade-off involves deciding upon a fixed- or floating-point representation. Fixed-point offers the minimum hardware requirements and, therefore, exhibits the lowest power consumption of the two. Unfortunately, it also suffers the most from dynamic range limitations and must be incorporated into the processor micro-code, which results in some runtime overhead. Floating-point, in contrast, alleviates the dynamic range difficulties at the expense of extensive hardware additions. This increased hardware leads to correspondingly higher capacitances and power consumption. As a result, floating-point should be selected only when absolutely required by dynamic range considerations. Decisions involving selection of the word length for the datapath and the accuracy of the floating point should be taken after a careful analysis of the requirements of an application, rather than the desired for an efficient low-power implementation. Aside from issues of accuracy and word length, the designer must also select an arithmetic representation for the data. For example, two's-complement, sign-magnitude, and canonical signed-digit are all possible arithmetic representations for data. Two's-complement is the most amenable to arithmetic computation and, therefore, is the most widely used. In this representation, the least significant bits (LSB's) are data bits, while the most significant bits (MSB's) are sign bits. As a result, the

MSB's contain redundant information, which can lead to wasted activity (and power) for data signals that experience a large number of sign transitions. In contrast, sign-magnitude data uses a single bit to describe the sign of the data and so sign transitions cause toggling in only one bit of the data [27]. A related issue is that of data encoding. Logarithmic companding can be used instead of floating-point to achieve similar results. Unfortunately, as with sign-magnitude, many computations (such as additions) don't have straightforward implementations in the logarithmic domain; however, some computations actually become simpler and less power consuming in this domain, e.g., multiplications translate to additions. Applications requiring a large number of multiplications can take advantage of this fact by using logarithmically encoded data. Clearly, there are many trade-offs involved in selecting a data representation for low-power systems. It is unlikely that any one choice would be ideally suited for all applications. Instead, a careful analysis of the application requirements in terms of performance and accuracy should be done before selecting the appropriate representation. Moreover, it might be beneficial to use different data representations in different parts of the systems at the expense of some data conversion overhead.

2.9 ALGORITHM LEVEL

Algorithmic-level power reduction techniques focus on minimizing the number of operations, weighted by the cost of those operations. Selection of an algorithm is generally based on details of an underlying implementation such as the energy cost of an addition versus a logical operation, the cost of a memory access, and whether locality of reference, both spatial and temporal can be maximized. The presence and structure of cache memory, for example, may cause a different set of operations to be selected, since the cost of a

memory access, relative to that of an arithmetic operation, changes. In general, reducing the number of operations to be performed is a first-order goal, although in some situations, recomputation of an intermediate result may be cheaper than spilling to and reloading from memory. Techniques used by optimizing compilers, such as strength reduction, common subexpression elimination, and optimizations to minimize memory traffic are also useful in most circumstances in reducing power. Loop unrolling may also be of benefit, as it results in minimized loop overhead as well as the potential for intermediate result reuse. Number representations offer another area for algorithmic power trade-offs. For example, the choice of using a fixed point or a floating-point representation for data types can have a significant difference in power consumption during arithmetic operations. Selection of sign-magnitude versus two's complement representation for certain signal processing applications can result in significant power reduction if the input samples are uncorrelated and dynamic range is minimized [28]. Operator precision, or bit length, is another trade off that can be selected to minimize power at the expense of accuracy. For some floating point algorithms, full precision can be avoided, and mantissa and exponent width reduced below the standard 23 and 8 bits, respectively, for single precision IEEE floating point standard. In [103], the authors show that for an interesting set of applications involving speech recognition, pattern classification, and image processing, mantissa bit width may be reduced by more than 50% to 11 bits with no corresponding loss of accuracy. In addition to improved circuit delays, energy consumption of the floating point multiplier was reduced 20% - 70% for mantissa reductions to 16 and 8 bits, respectively. Truncation of low-order bits of partial sum terms when performing a 16-bit fixed-point multiplication has been shown to result in power savings of 30% due mainly to reduction in area [93]. Adaptive bit truncation techniques for

performing motion estimation in a portable video encoder are shown to save 70% of the power over a full bit width implementation [45].

2.10 SUMMARY

In previous sections we discussed the mechanisms of power dissipation. We discussed various existing techniques of power reduction at different abstraction levels ranging from layout and technology to architecture and system and outlined the benefits and limitations of those techniques.

Chapter 3

Hardware - Software Technique using Pipeline stalls to reduce leakage power

In the previous chapter we discussed the mechanisms of power dissipation and various techniques used at different abstraction levels to reduce power. In this chapter we describe a new Architecture and System level technique to reduce leakage power.

3.1 Hardware-Software Technique

We already discussed the impact of the technology scaling on power dissipation in Chapter 1. In particular, due to the scaling down of the threshold voltage, an exponential growth in subthreshold leakage current is expected with every cranking of the technology wheel [21]. Similarly, scaling down of the gate geometry (and in particular, oxide thickness) is resulting in a very rapid growth of the gate leakage current [42]. Without corrective measures at the device, circuit and/or microarchitecture level, the total standby (leakage) power may well become the dominant part of the total power consumed by a microprocessor chip in the future technologies.

To solve this problem of increasing leakage power consumption in the high-leakage technologies we have proposed a hardware-software technique to reduce leakage power of microprocessors at the architecture level. We present a simulated experiment to evaluate this technique for a pipelined processor.

A simple and obvious way to reduce power consumption is to decrease the clock frequency f. Decreasing f causes a proportional decrease in the dynamic power dissipation.

The power consumption over a given period of time is reduced but slowing the clock also results in slower computations, hence the rate of useful work done is reduced and the system then operates for a longer period of time to execute the given task. Two of the limiting constraints of the clock frequency reduction technique are throughput and peak-performance. So for the peak-performance constrained and throughput constrained systems, clock frequency reduction is not a viable alternative for power optimization.

As the clock frequency is reduced, the amount of work done in a given period of time is reduced and this leads to reduction in the dynamic power but the leakage power remains unchanged. So, for the future technologies where leakage power is of concern this will lead to very high leakage energy dissipation. In order to reduce this leakage power dissipation, instead of reducing the clock frequency of the processor we reduce the execution rate of the processor by inserting NOPs in the pipeline after every instruction. According to the performance needed we can add as many NOPs after each instruction as is possible. This does affect the throughput of the processor and so can be applied to processors when they are not throughput constrained. This technique is more efficient in the case where the programs executed on the normal processor have lots of hazards. Such programs will take longer time to execute in the normal processor and will need to add bubbles in the pipeline to remove the hazards, but in the case where the NOPs are added after every instruction, many hazards will get resolved and the execution time will be less compared to the normal processor.

Once a NOP is inserted in the pipeline the control unit decodes the NOP instruction and generates power signals for the hardware components of the processor that consume most power and puts them into sleep mode. In the sleep mode the power supply is either

completely or partially cut off. The power signals thus allow significant saving of power during the cycles when NOPs are being executed.

Power-gating is a technique for reducing leakage power by shutting off the idle blocks. Implementation of power-gating requires a multi-threshold CMOS process. Logic blocks are implemented using low-Vt, high-performance transistors whereas high-Vt transistors (called sleep transistors) connect the gated blocks to the power supply [78].

Microarchitectural technique for power gating of the Execution Units is explained in [51]. In that paper, parameterized analytical equations that estimate the break-even point for application of power-gating techniques are first developed. The potential for power gated execution units is then evaluated, for the range of relevant break-even points determined by the analytical equations, using a state-of-the-art out-of-order superscalar processor model. The power gating potential of the floating-point and fixed-point units of this processor is then evaluated using three different strategies to detect opportunities for entering sleep mode, namely, ideal, time-based, and branch-misprediction-guided.

In the ideal technique, power gating is achieved by using a suitably sized header (Fig. 3.1) or footer transistor for a circuit block that is deemed to be a power-gating candidate. When the logic detects the onset of a sufficiently long idle period of a target circuit block, a "sleep" signal is applied to the gate of the header or footer transistor to turn-off the supply voltage of the circuit block. Similarly, once it is determined that the circuit block is being requested for use, the "sleep" signal is de-asserted to restore the voltage at the virtual Vdd.

In the time-based technique, the execution unit is power-gated after observing predetermined number of idle cycles and restarting the execution unit with a performance penalty once a pending operation is detected.

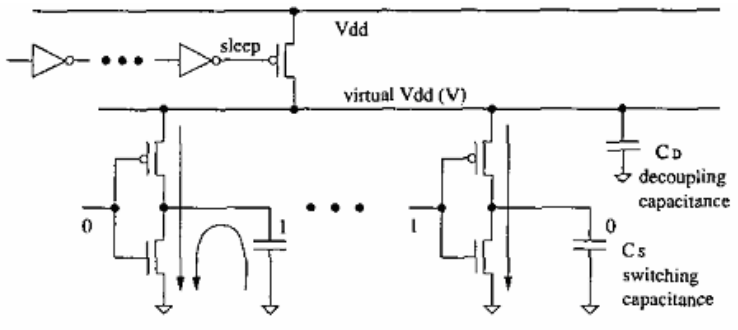

Figure 3.1: Using Headers for Power Gating

The other technique given by the author [78], the branch prediction mechanism guides the gating of execution units. An execution unit is turned off as soon as the branch misprediction is detected.

The results show that using the time-based approach, floating-point units can be put to sleep for up to 28% of the execution cycles at a performance loss of 2%. For the more difficult to power-gate fixed-point units, the branch misprediction guided technique allows the fixed-point units to be put to sleep for up to an additional 40% of the execution cycles compared to the simpler time-based technique, with similar performance impact.

For the caches, the preservation of cache states during standby mode is often desirable, which means it would be good if data stored in caches were not destroyed so that we won't need to access secondary memories on recovery. The other thing is that memory access time should not be greatly degraded, which means recovery time should be as small as possible, otherwise it will severely compromise the system performance. Two most widely cited methods are decay cache and drowsy cache [64].

Decay cache utilizes the gated-Vdd technique. This technique reduces the leakage power by using a high threshold (high-Vt) transistor to turn off the power to the memory cell when

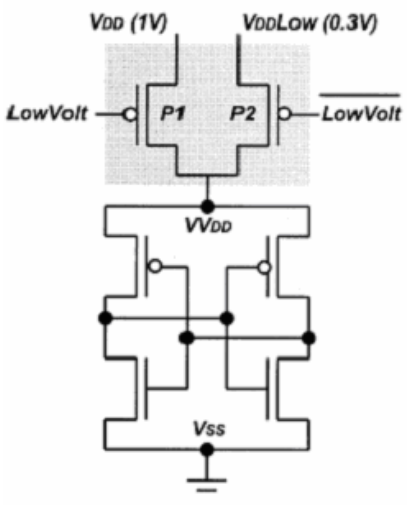

Figure 3.2: Supply Voltage Control Mechanism

the cell is set to low-power mode. This high-Vt device drastically reduces the leakage of the circuit because of the exponential dependence of leakage on Vt. While this method is very effective at reducing leakage, its main disadvantage lies in that it loses any information stored in the cell when switched into low-leakage mode. This means that a significant performance penalty is incurred when data in the cell is accessed and more complex and conservative cache policies must be employed. This technique reduces energy-delay by 62% with minimal impact on performance.

Drowsy cache, as described by Kim et al. [55], provides a better solution. It also uses transistors to separate virtual Vdd from Vdd supply line but still supplies a very low voltage to the cell when it is turned into low power mode. The cell implementation is shown in Figure 3.2.

According to the authors [55], the wake up latency is only a few clock cycles and thus does not have a major impact on the system performance. For data caches, all cache lines except the active one are put into drowsy mode every n clock cycles. The integer n depicts

the window size of how often should the cache be put into drowsy mode and they found 4000 is an adequate number for the benchmark they run on. Since programs typically only access a small portion of the entire data in the memory, the drowsy cache method could gain a significant reduction in leakage power consumption in the long run. By this method 80%-90% of the data cache lines can be maintained in a drowsy state without affecting the performance by more than 0.6%, even though moving lines into and out of a drowsy state incurs a slight performance loss.

For instruction caches, the situation is slightly different due to the instruction access characteristics. Therefore putting all cache lines into drowsy mode every n cycles does not work well for instruction caches. However, the spatial locality property can still be utilized. Kim et al. [55] proposed a low leakage instruction cache architecture based on the subbank method. The basic idea of subbank is to divide the cache into several subbanks and turn those inactive subbanks into low power mode. The proposed architecture extends the subbank method and adds Next Subbank Prediction Techniques to it. A prediction buffer keeps track of predicted subbank index and other information for the instruction fetched one cycle earlier. Thus if the instruction (e.g., a jump instruction) is going to access a subbank in drowsy mode, that subbank could be woken up one cycle earlier to enhance the performance. There are also other techniques for instruction caches such as the one described by Kalla et al. [53]. The authors perceived that programs, especially multimedia applications, tend to spend most of their time in loops and execute only a sequence of instructions for most of their computations. Based on this observation, they propose a novel cache replacement policy for instruction caches, which forces instructions in a loop to be placed in the same subbank and are not the first candidates to be replaced into secondary

memories when misses occur. In such a way, only one subbank will stay active and other subbanks can stay in the drowsy mode most of the time.

Power-aware compilation for register file energy reduction has been discussed in a recent paper [11].

Power-gating techniques as mentioned above can be applied to the different components of the processor which are not in use when the NOP instruction is inserted into the pipeline of the processor.

Power gating techniques are discussed by other authors as well [4, 34].

3.2 Conclusion

A software approach of inserting NOPs after every instruction, combined with a hardware approach of power-gating the processor components that are not in use when the NOP is executed, is explained in this chapter to gain the leakage power savings. Because clock period is not changed for the non-NOP instructions, we have not assumed any reduction in the supply voltage, which would have slowed down the hardware. However, there has been reported work on speculative hardware speed reduction for power saving [38]. Here, voltage reduction may cause some errors, which are detected and the instructions are reexecuted. As long as the error rate is small, one can obtain power saving with negligible performance penalty. Using such procedures with the NOP insertion may be investigated in the future.

Chapter 4

Theoretical and Experiment Results

In the previous chapter, a hardware-software technique for reducing the leakage power was explained. The theoretical results as well as practical results are discussed in this chapter. The technique was applied to a 32-bit MIPS pipelined processor using CMOS circuitry. We assumed the Berkeley Predictive Technology Models for 65nm and 22nm CMOS technologies [2]. Though this demonstration uses one particular processor, the technique can be applied to other processors as well.

4.1 THEORETICAL RESULTS

4.1.1 Clock Slow-Down Method

As we discussed in the previous chapter we are using the clock frequency reduction method as our reference method. To reduce power when we slow down the clock, dynamic power is reduced in proportion to clock rate whereas leakage power remains unchanged. However, the computing task now takes longer to complete. This results in the same dynamic energy consumption whereas the leakage energy consumed is more. For normal operation, we assume:

Rated clock frequency as: f

Dynamic power as: Pd

Static power as: Ps

Then,

$$Total\ power\ consumed\, P(1) = Dynamic\ power + Static\ power = Pd + Ps \tag{4.1}$$

and,

$$Energy\ consumed\ by\ an\ N-cycle\ task\, E(N,1) = Power \times Time = (Pd+Ps)N/f \tag{4.2}$$

For power saving mode where the clock frequency is reduced by factor n:

Clock frequency = f/n

Dynamic Power = Pd/n, as dynamic power is reduced in proportion to clock rate

Static Power = Ps, as static power remains unchanged

Therefore, in this case,

$$Total\ power\ consumed\, P(n) = Pd/n + Ps \tag{4.3}$$

and,

$$Energy\ consumed\ by\ an\ N-cycle\ task\, E(N,n) = (Pd + nPs)N/f \tag{4.4}$$

Using equations 4.1 and 4.3, the power saving ratio is obtained as follows:

$$P-ratio = P(1)/P(n) \tag{4.5}$$

$$P-ratio = n(Pd+Ps)/(Pd+nPs) \tag{4.6}$$

$$P-ratio = n(k+1)/(k+n)\ \ where\ k = Pd/Ps \tag{4.7}$$

For low leakage technologies where static power consumption is negligible compared to dynamic power consumption, k >> 1. In this case,

$$P - ratio = n \tag{4.8}$$

For high leakage technologies where static power consumption is higher and of concern, assuming $k \leq 2$, power ratio we obtain for different values of k is as follows:

$$P - ratio = 3n/(n+2)\ for\ k = 2 \tag{4.9}$$

where $k = 2$ means dynamic power consumed is double the static power consumption. Further,

$$P - ratio = 2n/(n+1)\ for\ k = 1 \tag{4.10}$$

where $k = 1$ means dynamic power consumed is equal to the static power consumption. Also,

$$P - ratio = 3n/(2n+1)\ for\ k = 0.5 \tag{4.11}$$

where $k = 0.5$ means dynamic power consumed is half of the static power consumption. These results are plotted in the graph shown in Figure 4.1.

From Figure 4.1, we observe that for low-leakage technologies, the power saving obtained is linear with the clock slow-down factor n but as the static power increases in proportion to the dynamic power, as is the case for the future high-leakage technologies, the power savings obtained by this method shall reduce.

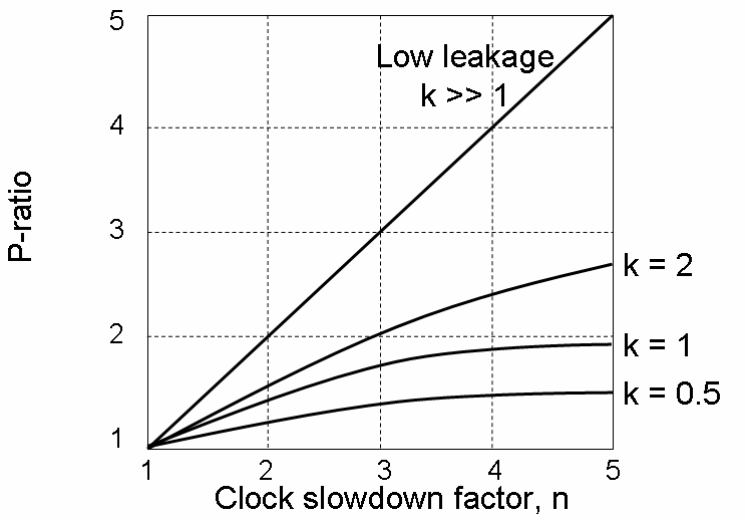

Figure 4.1: Power Saving Ratio for Clock Slow-Down Method

Similarly, calculating energy saving ratio from equations 4.2 and 4.4 we get:

$$E - ratio = E(N,1)/E(N,n) \tag{4.12}$$

$$E - ratio = (Pd + Ps)/(Pd + nPs) = n \times P - ratio \tag{4.13}$$

$$E - ratio = (k+1)/(k+n) \;\; where \; k = Pd/Ps \tag{4.14}$$

For low leakage technologies where static power consumption is negligible compared to dynamic power consumption, $k >> 1$. In this case,

$$E - ratio = 1 = constant \tag{4.15}$$

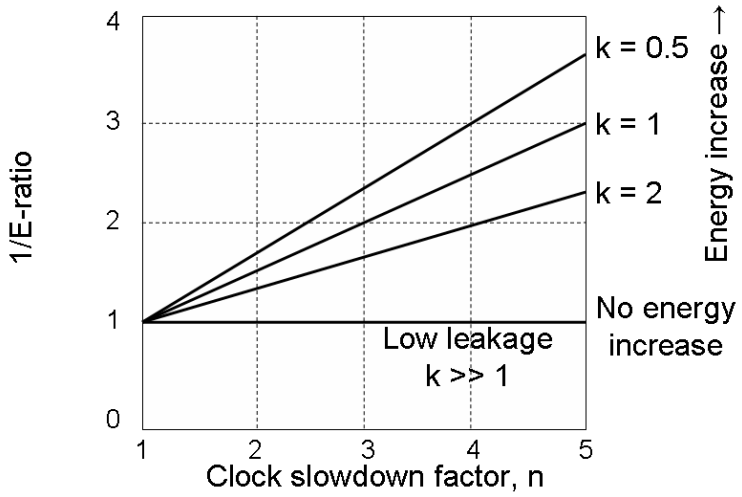

Figure 4.2: Energy Saving Ratio for Clock Slow-Down Method

For high leakage technologies where static power consumption is higher and of concern, assuming $k \leq 2$, energy ratio we obtain for different cases of k is as follows:

$$E - ratio = 3/(n+2)\ for\ k = 2 \tag{4.16}$$

where $k = 2$ means dynamic power consumed is double the static power consumption. Further,

$$E - ratio = 2/(n+1)\ for\ k = 1 \tag{4.17}$$

where $k = 1$ means dynamic power consumed is equal to the static power consumption. Also,

$$E - ratio = 3/(2n+1)\ for\ k = 0.5 \tag{4.18}$$

where $k = 0.5$ means dynamic power consumed is half of the static power consumption. These results are plotted in the graph shown in Figure 4.2.

From Figure 4.2, we observe that for low-leakage technologies, there is no increase in energy and hence the energy saving obtained is constant but as the static power increases in comparison to the dynamic power, as for the future high-leakage technologies, the energy consumption will go on increasing.

4.1.2 Instruction Slow-Down Method

To differentiate from the clock slow-down methodology, we will call the NOP insertion as the *instruction slow-down* method. In this new energy saving method the rated clock frequency (f) is maintained. Power management hardware inserts nops after each instruction. Let this instruction slowdown factor be m, where $m = 0$ for normal operation. Once the nops are inserted the management unit provides hardware sleep modes to reduce NOP power. Power control signals are generated by control logic for each individual unit of the processor to set them into their individual low-power modes as discussed in Chapter 3. To analyze this technique, let

P = power consumed by instruction cycles,

P/f = energy consumed per instruction cycle,

$\beta P/f$ = energy consumed per NOP cycle,

where β = reduction factor ($0 \leq \beta \leq 1$) due to power down/sleep modes.

Hence, for this new technique, for a given time period as illustrated in Figure 4.3, we get

$$Power = P(1 + m\beta)/(m + 1) \tag{4.19}$$

For the normal operation mode in the new instruction slow-down method we have,

Rated clock frequency, f and $m = 0$ (as no NOP is inserted)

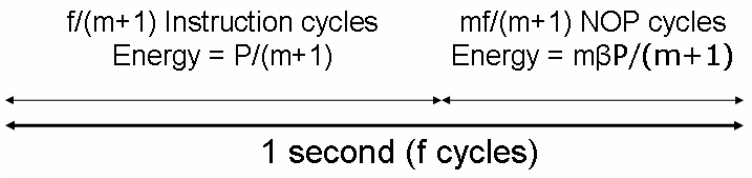

Figure 4.3: Distribution of Instruction Energy and NOP Energy for a given time period

Assuming,

Dynamic power: Pd

Static power: Ps

$$Total\ power\ consumed = Pd + Ps \tag{4.20}$$

and,

$$Energy\ consumed\ by\ N - cycle\ task = (Pd + Ps)N/f \tag{4.21}$$

For the power saving mode where the rated clock frequency f is maintained, using equation 4.19, we get

$$Dynamic\ Power = Pd(1 + m\beta)/(m + 1) \tag{4.22}$$

$$Static\ Power = Ps(1 + m\beta)/(m + 1) \tag{4.23}$$

Hence,

$$Total\ power\ P(m) = (Pd + Ps)(1 + m\beta)/(m + 1) \tag{4.24}$$

However, now a given N-cycle task will take longer to complete as NOPs are inserted after each instruction. The energy consumed by the N-cycle task is given by,

$$E(N, m) = (Pd + Ps)[(1 + m\beta)/(m + 1)]N(m + 1)/f = (Pd + Ps)(1 + m\beta)N/f \tag{4.25}$$

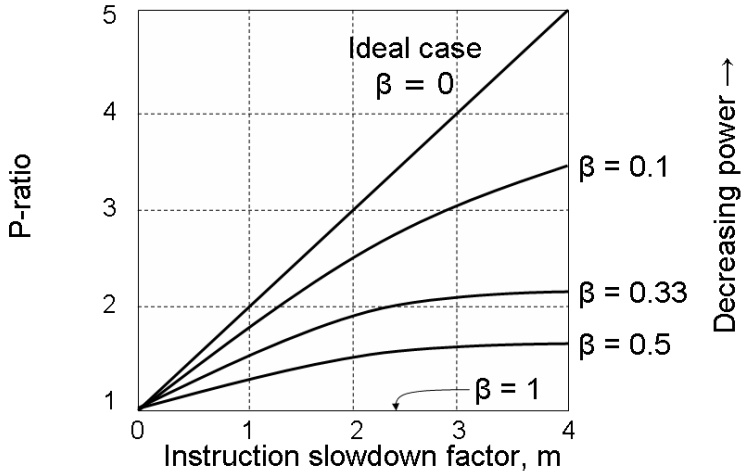

Figure 4.4: Power Saving Ratio for Instruction Slow-Down Method

From equations 4.20 and 4.24 the power saving ratio for the instruction slow-down method can be obtained as:

$$P - ratio = P(0)/P(m) \tag{4.26}$$

$$P - ratio = (m+1)/(1+m\beta) \tag{4.27}$$

The plot of the $P-ratio$ as obtained from equation 4.27 is shown in Figure 4.4. For the case $\beta = 1$, where the NOP cycle consumes the same power as the instruction cycle we see no power saving and this method is not effective. When the NOP cycle consumes less power compared to an instruction cycle, we observe power savings. For the case where $\beta = 0$, which is the ideal case where NOP cycle consumes no power at all, we observe the power saving linear with the instruction slow-down factor m.

Next, from equations 4.21 and 4.25, we get the energy saving ratio as follows:

$$E - ratio = E(N,0)/E(N,m) \tag{4.28}$$

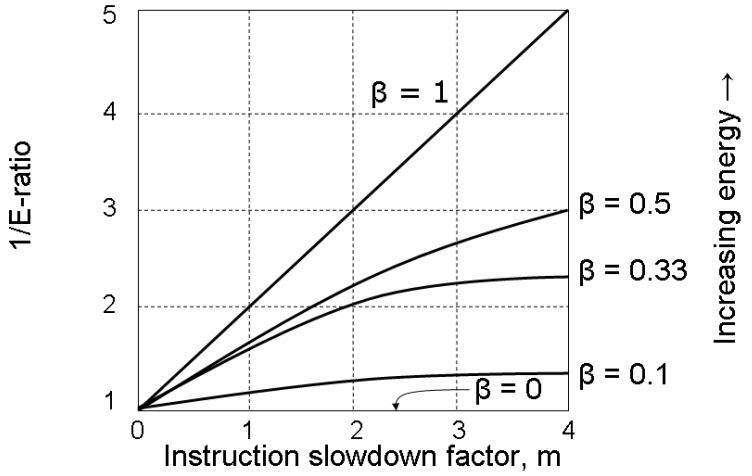

Figure 4.5: Energy Saving Ratio for Instruction Slow-Down Method

$$E - ratio = 1/(1 + m\beta) \tag{4.29}$$

A plot of the $E - ratio$, obtained by equation 4.29, is shown in Figure 4.5. For the case $\beta = 1$, where the NOP cycle consumes the same power as the instruction cycle we see the energy consumed increases linearly with the instruction slow-down factor m. However, when the NOP cycle consumes less power compared to instruction cycle we observe that the energy consumption decreases. For $\beta = 0$, which is the ideal case where NOP cycle consumes no power at all, we observe that there is no increase in energy. Hence, for the ideal case, where NOP cycle consumes no power, there is no increase in the energy whereas the power saving is linear to the instruction slow-down factor m.

4.1.3 Comparison of Clock Slow-Down and Instruction Slow-Down Methods

From equations 4.4 and 4.25, we compute the energy ratio comparing the two methods as follows:

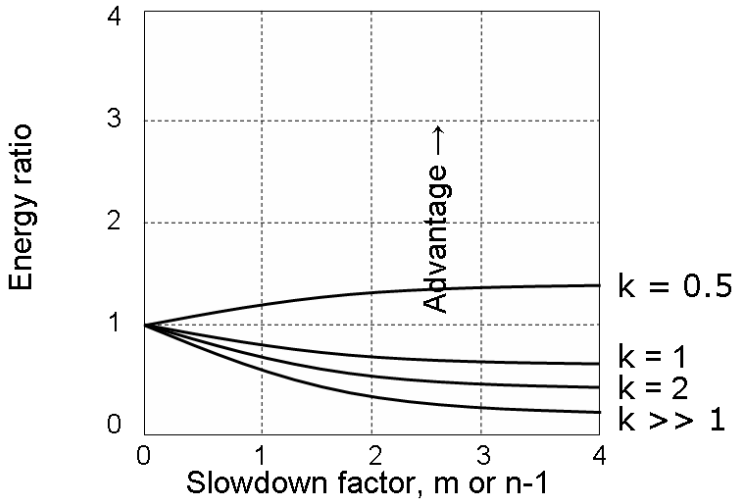

Figure 4.6: Clock Slowdown Method Vs. Instruction Slowdown Method for $\beta = 1$ (No Sleep Mode)

$$Energy\ (Clock\ slowdown)/Energy\ (Instruction\ slowdown) = (k+m+1)/[(k+1)(1+m\beta)] \tag{4.30}$$

where, $n = m + 1$ and $k = Pd/Ps$. This ratio can be plotted for different values of β as shown in Figures 4.6, 4.7 and 4.8.

From Figure 4.6 we observe that for $\beta = 1$, when there is no sleep mode applied and the NOP cycle consumes the same power as instruction cycle, the new instruction slow-down technique is not too efficient for high-leakage technologies and it provides no benefit for the low-leakage technologies. Instruction slow-down shows advantage for $k = Pd/Ps = 0.5$, i.e., when static power is twice that of dynamic power, but that is only because of the inefficiency of the clock slow-down method.

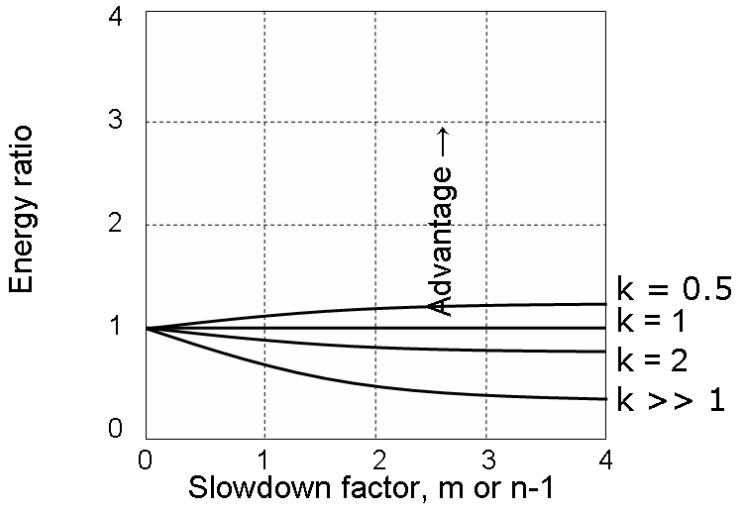

Figure 4.7: Clock Slowdown Method Vs. Instruction Slowdown Method for $\beta = 0.5$ (Sleep Mode)

For $\beta = 0.5$, where NOP cycle consumes 50% less power than the instruction cycle, in Figure 4.7 we observe a break-even point for the case where $k = Pd/Ps = 1$. That is, for a technology where dynamic power is equal to the static power, the instruction slow-down technique shows the same energy consumption as the clock slow-down technique. For this case too the instruction slow-down technique provides no significant benefit for low-leakage technologies.

For $\beta = 0.1$, where NOP cycle consumes 90% less power than the instruction cycle, in Figure 4.8 we observe significant advantage from instruction slow-down except when leakage is very small ($k = Pd/Ps >> 1$).

Hence, the instruction slow-down technique is more efficient for high-leakage technologies than the clock-slow down technique for the case where NOP cycle consumes 50% or less power than the normal instruction cycle.

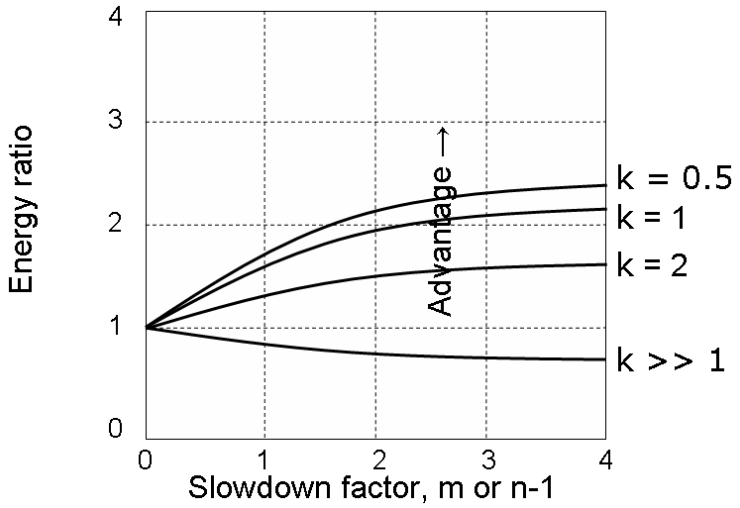

Figure 4.8: Clock Slowdown Method Vs. Instruction Slowdown Method for $\beta = 0.1$ (Sleep Mode)

The new technique was applied to a 32-bit MIPS processor [8, 9]. The architecture of the processor is discussed in next section.

4.2 MIPS PROCESSOR

The MIPS architecture [46, 80] is a widely supported processor architecture, with a vast infrastructure of industry-standard tools, software and services that help ensure rapid, reliable and cost-effective system-on-chip (SoC) design. The MIPS processor, designed in 1984 by researchers at Stanford University, uses a RISC (Reduced Instruction Set Computer) instruction set. Compared with their CISC (Complex Instruction Set Computer) counterparts (such as the Intel's Pentium processors), RISC processors typically support fewer and much simpler instructions.

Table 4.1: MIPS Instruction Formats

Format	Bits 31-26	Bits 25-21	Bits 20-16	Bits 15-11	Bits 10-6	Bits 5-0
R	op	rs	rt	rd	shamt	funct
I	op	rs	rt	imm		
J	address					

4.2.1 MIPS Instruction Formats

The meanings of the fields in MIPS instructions presented in Table 4.1 are as follows:

op : opcode. basic operation of the instruction.

rs : the first register source operand.

rt : the second register source operand.

rd : the register distination operand. It gets the result of the operation.

shamt : shift amount.

funct : function. This field selects the specific variant of the operation in the op field.

A compromise choice made by the MIPS designers is to keep all instructions the same length, thereby requiring different kinds of instruction formats for different kinds of instructions. The format above is called R-type (for register), I-type (for immediate), and J-type(for jump).

For the particular processor used for the experiment the supported instructions are load word (LW), store word (SW), add (ADD), subtract (SUB), branch on equal (BEQ), jump (J), and no operation (NOP). LW and SW use the immediate-format (I-format); the operands are the destination/source register address, the register address storing the base memory address, and an immediate value for the data memory address offset. BEQ also follows the I-format such that it takes two register addresses to test for equality and an

Table 4.2: Format and the meaning of the supported Instructions

Name	6 Bit	5 Bit	5 Bit	5 Bit	5 Bit	6 Bit	Assembly	Meaning
lw	35	2	1	100			lw$1, 100($2)	$1 <= DMem[$2+100]
sw	43	2	1	100			sw$1, 100($2)	DMem[$2+100] <= $1
add	0	2	3	1	0	32	add $1, $2, $3	$1 <= $2 + $3
sub	0	2	3	1	0	34	sub $1, $2, $3	$1 <= $2 - $3
beq	4	1	2	25			beq $1, $2, 100	if($1 == $2), then PC <= PC+4+100
j	2	2500					j 10000	PC <= 10000
nop	0						nop	Do Nothing

immediate value to add to the program counters value should the equality test pass. ADD and SUB follow the register-format (R-format) where the operands are the destination register address and two source register addresses. Finally, J uses the jump-format (J-format); its operand is an immediate value to store into the program counter. Table 4.2 summarizes the instruction formats.

4.2.2 Architecture

The processor datapath has five stages: instruction fetch (IF), instructions decode (ID), execute (EX), data memory (M), and write-back (WB).

INSTRUCTION FETCH: The IF stage involves keeping track of the current/next instruction as well as retrieving the current instruction from memory. In this scenario, memory is split into separate instruction and data memories in order to avoid a structural hazard. That is, simultaneous access to memory, one for instructions and the other for data, is possible in the architecture shown in Figure 4.9.

INSTRUCTION DECODE: In the next cycle, the fetched instruction moves into the ID stage. There, the instruction is broken up into several fields and inputs into the control logic and register file. Various control signals, register values, and intermediate values are

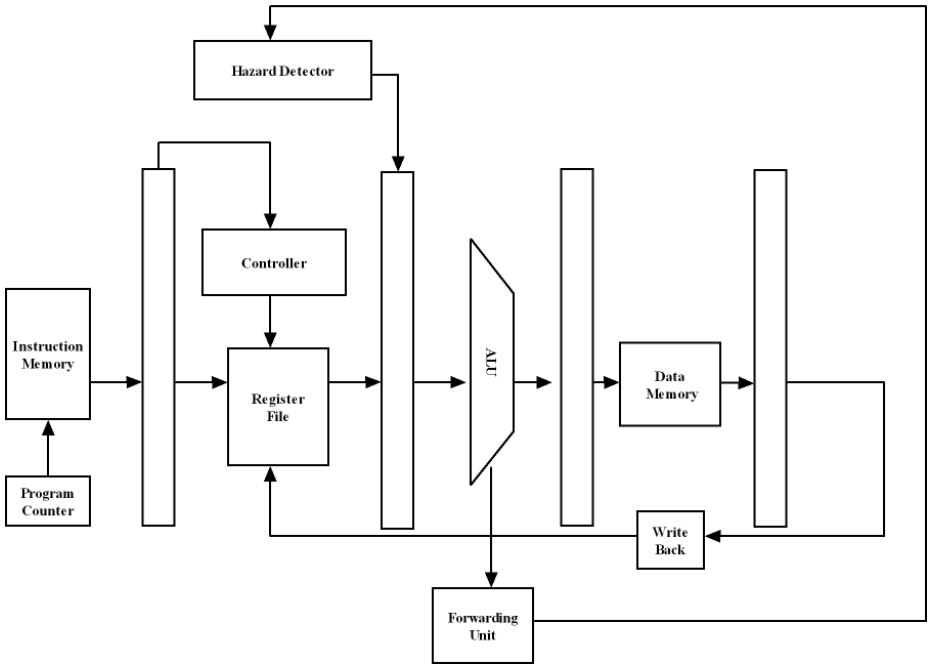

Figure 4.9: General Architecture of the Processor

handed to the EX stage where arithmetic operations are performed (in this case, integer add and subtract). In addition, the register addresses will be forwarded to the hazard detector in this stage. If there is potential hazard in the system, this stage will perform a stall.

EXECUTE: This is the main stage where most of the ALU operations are performed. Also, this is where the registers addresses are forwarded back to the ID stage for hazard detection.

DATA MEMORY: In the M stage, data is retrieved and/or stored into memory.

WRITE BACK: Finally, in the WB stage, applicable control signals and results, either from data memory or arithmetic calculations, are fed back to the register file.

4.2.3 Pipeline Hazards

There are situations, called hazards, that prevent the next instruction in the instruction stream from being executing during its designated clock cycle. Hazards reduce the performance from the ideal speedup gained by pipelining. There are three classes of hazards:

Structural Hazards: They arise from resource conflicts when the hardware cannot support all possible combinations of instructions in simultaneous overlapped execution.

Data Hazards: They arise when an instruction depends on the result of a previous instruction in a way that is exposed by the overlapping of instructions in the pipeline.

Control Hazards: They arise from the pipelining of branches and other instructions that change the PC.

Hazards in pipelines can make it necessary to stall the pipeline. The processor can stall on different events:

Table 4.3: A Sequence of Instructions

AND	**R1 , R2 , R3**
SUB	R4 , R5 , **R1**
AND	R6, **R1 , R7**

A cache miss: A cache miss stalls all the instructions on pipeline both before and after the instruction causing the miss.

A hazard in pipeline: Eliminating a hazard often requires that some instructions in the pipeline to be allowed to proceed while others are delayed. When the instruction is stalled, all the instructions issued later than the stalled instruction are also stalled. Instructions issued earlier than the stalled instruction must continue, since otherwise the hazard will never clear.

The problem with data hazards, introduced by the sequence of instructions as shown in table 4.3, can be solved with a simple hardware technique called forwarding. The key insight in forwarding is that the result is not really needed by SUB until after the ADD actually produces it. The only problem is to make it available for SUB when it needs it. If the result can be moved from where the ADD produces it (EX/MEM register), to where the SUB needs it (ALU input latch), then the need for a stall can be avoided.

Using this observation, forwarding works as follows:

The ALU result from the EX/MEM register is always fed back to the ALU input latches.

If the forwarding hardware detects that the previous ALU operation has written the register corresponding to the source for the current ALU operation, control logic selects the forwarded result as the ALU input rather than the value read from the register file.

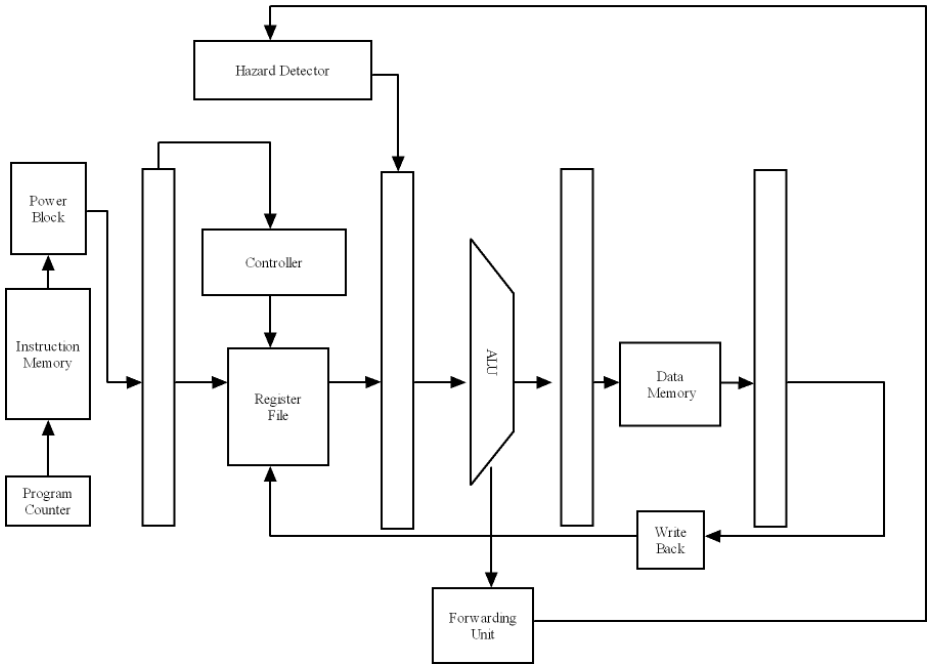

Figure 4.10: Modified Architecture of the Processor

4.3 MODIFIED MIPS PROCESSOR

The architecture of the given MIPS processor is modified in order to get low-power consumption. In the modified architecture, a Power Block is added in the fetch cycle of the Processor as shown in the Figure 4.10. This Power Block has an externally controlled 3-bit signal that is controlled by the user. According to the performance required, the 3-bit signal can be set to the number of NOPs that is inserted into the pipeline after every instruction. With the 3-bit signal, maximum of 7 NOPs can be inserted into the pipeline with the bit condition as shown in Table 4.4.

A schematic of the power block is shown in Figure 4.11.

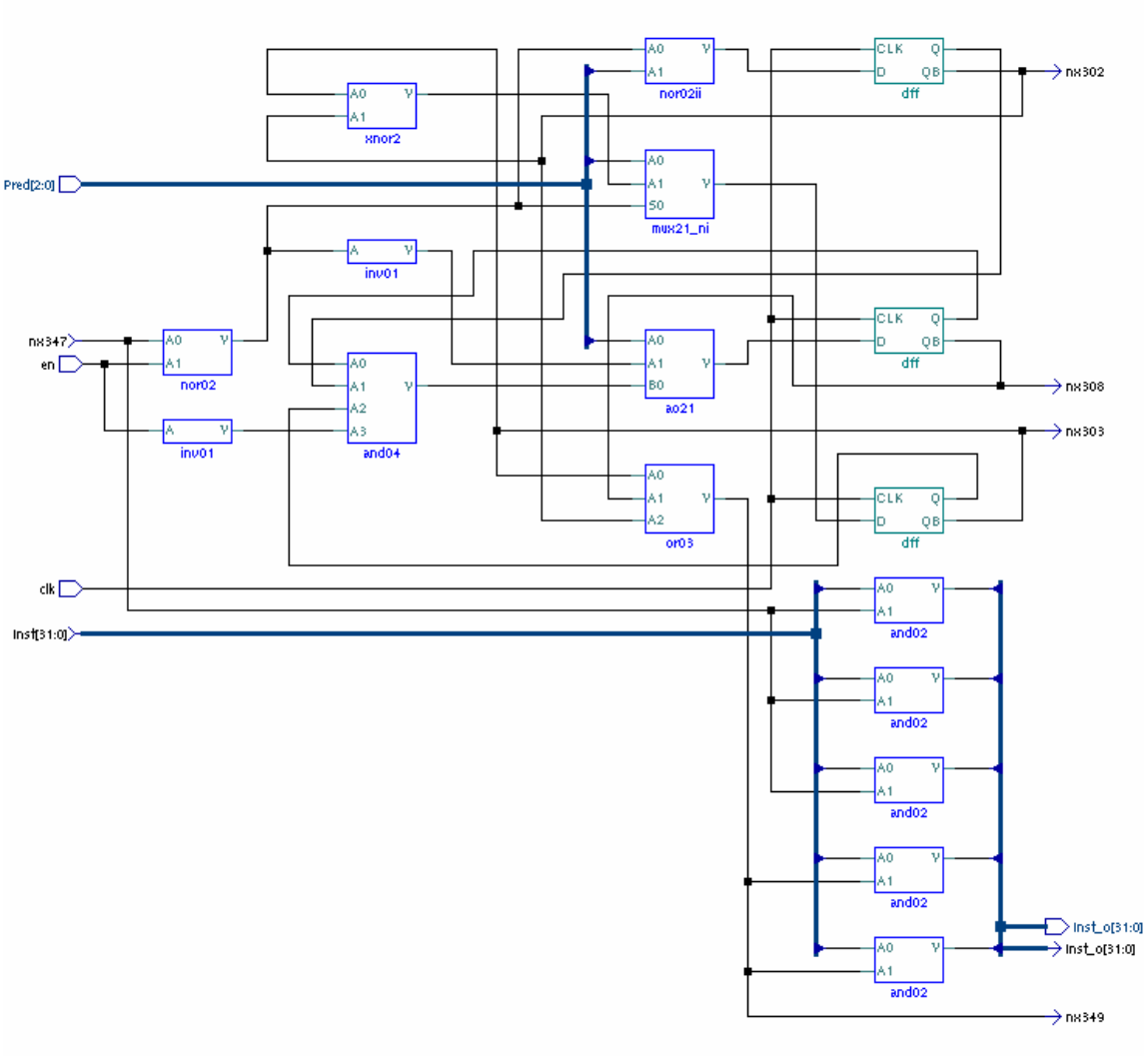

Figure 4.11: Schematic of the Power Block

Table 4.4: External 3-bit signal conditions

000	Normal Processor Operation
001	1 NOP inserted after every instruction
010	2 NOPS inserted after every instruction
011	3 NOPS inserted after every instruction
100	4 NOPS inserted after every instruction
101	5 NOPS inserted after every instruction
110	6 NOPS inserted after every instruction
111	7 NOPS inserted after every instruction

In this architecture the instruction memory instead of feeding to the pipeline register for the decode stage is fed to this Power Block. The Power Block then performs two operations. It reads the external instruction slowdown signal and decides whether the normal stream of instructions should be executed or the NOPs should be inserted in the pipeline after every instruction. The second operation it performs is to maintain the program counter pointing at the same instruction until the NOPs are executed. Once the NOPS are inserted into the pipeline the control unit decodes the NOP instruction and generates the power signals and puts the data-memory, register file and the ALU into low-power mode by power-gating techniques as discussed in Chapter 3. A low-power mode can also be applied to the instruction memory when the pipeline is executing the inserted NOP instruction. Once the NOPs are executed the program counter fetches the next instruction in instruction memory, which is then executed followed by NOPs again.

4.4 RESULTS

4.4.1 Blocks of the Original Processor

Firstly the power consumption of different blocks of the processor was found out in order to understand which block of the processor consumes the most power and needs to be put down into low-power mode. For getting the power estimation of these blocks the structural Verilog netlist of the blocks was taken and converted into the Rutger's mode gate-level netlist. This netlist was then estimated for power by the tool developed by Jins Alexander at Auburn University [7]. The power results for the blocks were obtained by applying 1000 random input vectors with the vector period of 100ns and rise time of 1ns for a combinational circuit and the vector period of 200ns with rise time of 1ns for sequential circuit. Operating voltage applied is 1V. Fanout wire load delay format was used with delay of each gate in ns. Each of these blocks was power estimated for two different technologies, 65nm and 22nm technology and the clock period used is 10ns. The results are in microwatts. The results obtained are shown in Tables 4.5 and 4.6.

From Tables 4.5 and 4.6 we observe that as the technology is scaled down to 22nm from 65nm, the leakage power increases and the dynamic power decreases. The leakage power is relatively high in 22nm technology and forms the major part of the total power consumption. We also observe that for this particular processor the data memory, register file and ALU consume more power and so power-gating techniques can be applied to these units to put them into low-power mode when the NOP is executed.

However, the results of Tables 4.5 and 4.6 show that the expected ratio of dynamic and leakage power is not maintained as the Berkeley Predictive Technology Model files available may have been modified to account for certain industry processes.

Table 4.5: Estimated power in microwatts for different blocks of the processor (65nm CMOS technology, clock period 10ns)

Block Name	Number of gates	Avg Leakage Power	Avg Dynamic Power	Total Avg Power
add-1-word	58	1.30247	0.371326	1.673802
add-nbits	142	3.310713737	1.322882	4.633595
alu	126	3.760219442	2.584336	6.344556
comparator	43	1.174578188	0.288572	1.463151
control-logic	12	0.221470117	0.111679	0.333149
data-mem	649	5566.433072	0.044306	5566.47731
inst-mem	63	0.99482736	0.061841	1.056669
hazard	30	0.754764244	0.17651	0.931275
forward	39	0.919369882	0.223469	1.142839
ex-m	73	3142.959671	4.311972	3147.271695
id-ex	123	5176.235456	6.456847	5182.692315
if-id	150	4201.080184	4.525968	4205.605946
m-wb	71	2997.391392	4.288376	3001.679666
pc	60	1690.834528	2.002712	1692.837221
regfile	4178	78306.37693	19.659017	78326.03902

Table 4.6: Estimated power in microwatts for different blocks of the processor (22nm CMOS technology, clock period 10ns)

Block Name	Number of gates	Avg Leakage Power	Avg Dynamic Power	Total Avg Power
add-1-word	58	61.57274038	0.287966	61.860708
add-nbits	142	160.1353579	0.678731	160.814088
alu	126	183.8257594	1.278279	185.104043
comparator	43	57.45039016	0.075063	57.525453
control-logic	12	10.12932989	0.074958	10.204288
data-mem	649	7573.0565	0.011231	7573.067676
inst-mem	63	54.12064638	0.016018	54.136664
hazard	30	35.5	0.047986	35.560068
forward	39	44.38890755	0.059332	44.448239
ex-m	73	8541.498333	3.325714	8544.824086
id-ex	123	14295.63016	4.974914	14300.60528
if-id	150	8512.405679	1.761546	8514.16681
m-wb	71	8259.585127	3.308923	8262.894116
pc	60	3808.628768	0.910683	3809.539368
regfile	4178	146416.0234	5.043241	146421.06

Table 4.7: Estimated power in microwatts for different blocks of the processor (65nm CMOS technology, clock period 20ns)

Block Name	Number of gates	Avg Leakage Power	Avg Dynamic Power	Total Avg Power
ex-m	73	3147.808369	2.225534	3150.033997
id-ex	123	5183.414556	3.332566	5186.747294
if-id	150	3977.928776	2.623281	3980.551846
m-wb	71	3001.879202	2.213355	3004.09249
pc	60	1686.125412	1.033658	1687.159063
regfile	4178	78049.01153	15.659671	78064.67265

Table 4.8: Estimated power in microwatts for different blocks of the processor (65nm CMOS technology, clock period 40ns)

Block Name	Number of gates	Avg Leakage Power	Avg Dynamic Power	Total Avg Power
ex-m	73	3150.349716	1.131009	3151.480807
id-ex	123	5187.182222	1.693599	5188.875832
if-id	150	3976.784647	1.333143	3978.117835
m-wb	71	3004.232887	1.12482	3005.357692
pc	60	1683.653332	0.525302	1684.178598
regfile	4178	78611.9774	8.016249	78619.99422

To examine the effect of clock frequency on the blocks of the processor we again apply 1000 random vectors to the sequential blocks of the processor with the vector period of 200ns and rise time of 1ns. Operating voltage applied is 1V. Fanout wire load delay format is used with delay of each gate in ns. Results were obtained for two different frequencies 20ns and 40ns. The results for 65nm technology are shown in Tables 4.7 and 4.8

From Tables 4.7 and 4.8 we observe that as the clock frequency is reduced to half the dynamic power is reduced by half, whereas the leakage power is not stable. Even though the dynamic power reduces to half, the leakage power being too high results in the total average power increasing with the reduction in clock frequency.

Table 4.9: Estimated power in microwatts for different blocks of the processor (22nm CMOS technology, clock period 20ns)

Block Name	Number of gates	Avg Leakage Power	Avg Dynamic Power	Total Avg Power
ex-m	73	8547.215723	1.716498	8548.93215
id-ex	123	14304.46375	2.567698	14307.03141
if-id	150	8355.727419	0.991802	8356.719278
m-wb	71	8264.964446	1.707831	8266.672492
pc	60	3804.834792	0.47003	3805.304877
regfile	4178	146260.038	4.038493	146264.0762

Table 4.10: Estimated power in microwatts for different blocks of the processor (22nm CMOS technology, clock period 40ns)

Block Name	Number of gates	Avg Leakage Power	Avg Dynamic Power	Total Avg Power
ex-m	73	8550.218306	0.872319	8551.090956
id-ex	123	14309.11012	1.304896	14310.4149
if-id	150	8356.056176	0.50403	8356.560022
m-wb	71	8267.787285	0.867914	8268.655278
pc	60	3802.842693	0.238868	3803.081578
regfile	4178	146664.9324	2.018337	146666.944

Similarly, the clock frequency effects for the blocks of the processor for 22nm technology are shown in Tables 4.9 and 4.10.

4.4.2 Power Block

For the modified processor the new block added in the architecture is the power block. The power results for this new block are obtained by applying 1000 random input vectors with the vector period of 200ns, rise time of 1ns and operating voltage of 1V. Fanout wire load delay format was used with delay of each gate in ns. Power is estimated for two different

technologies, 65nm and 22nm technology, and the clock periods used are 10ns, 20ns and 40ns. The results are in microwatts and are shown in Tables 4.11 and 4.12.

Table 4.11: Estimated power in microwatts for power block of the processor (65nm CMOS technology)

Number of gates	Clock period	Avg Leakage Power	Avg Dynamic Power	Total Avg Power
49	10	43.57989	6.246311	49.826202
49	20	44.52892	3.089169	47.618098
49	40	46.42725	1.512446	47.939706

Table 4.12: Estimated power in microwatts for power block of the processor (22nm CMOS technology)

Number of gates	Clock period	Avg Leakage Power	Avg Dynamic Power	Total Avg Power
49	10	335.59347	3.457777	339.051243
49	20	336.11856	1.712433	337.831007
49	40	337.27590	0.840803	338.116719

For the power block too we observe that the leakage power is more for 22nm technology compared to 65nm technology. We also observe that as the clock frequency is reduced the dynamic power reduces, whereas the leakage power increases and, leakage power being too high and a major fraction of the total power, the total power increases with reduction in clock frequency. Even this results contradicts our expectations as the leakage power increases instead of remaining stable with the reduction in the clock frequency.

4.4.3 Comparison between the original and modified processors

When the power block is added into the architecture, and comparing it with the original processor when just 1 NOP is inserted after every instruction into the processor, the results

Table 4.13: Comparing the original and modified processors for 22nm technology for 1 NOP (power in microwatts)

Processor	No. of gates	Clock Period	Avg Leakage Pwr	Avg Dynamic Pwr	Total Avg Pwr
Original	6684	20ns	235042.2	3.352106	235045.5521
Modified	6787	10ns	225510.6	3.533928	225514.1339
Original	6684	40ns	237590.5	1.670603	237592.1706
Modified	6787	20ns	226043.6	1.746964	226045.347

Table 4.14: Comparing the original and modified processors for 65nm technology for 1 NOP (power in microwatts)

Processor	No. of gates	Clock Period	Avg Leakage Pwr	Avg Dynamic Pwr	Total Avg Pwr
Original	6684	20ns	116676.4	13.29557	116689.6956
Modified	6787	10ns	111695.1	14.057876	111709.1579
Original	6684	40ns	118611.3	6.529785	118617.8298
Modified	6787	20ns	113055.2	6.828938	113062.0289

obtained for 22nm technology are as shown in Table 4.13 and the results obtained for 65nm technology are as shown in Table 4.14.

From Tables 4.13 and 4.14 we observe that an average of about 4.46% of power savings is obtained for the modified processor when 1 NOP is inserted into the pipeline after each instruction. The power savings obtained from these results is not as much as expected. This might be because the technology files used for the experiment were not reliable. Also the tool used to obtain the results was not capable enough to simulate large designs and operated only for constant inputs whereas the design required pulsed inputs.

4.5 Summary

This chapter discusses the theoretical results, which show that the new energy saving instruction slow-down technique is better than the reference clock slow-down technique for higher leakage technologies for the case where NOP cycle consumes 50% or less power than the normal instruction cycle. The architecture of a 32-bit MIPS processor and the modified architecture of the new processor are also explained in this chapter. However, the practical results obtained were not as expected because of the unreliable technology files used for the experiment and the tool used to simulate the design was not efficient.

Chapter 5

Conclusion

Leakage power is a major concern in current and future microprocessor designs, since as with the technology scaling the leakage power is becoming a major component of the total power consumption. In this thesis, to reduce this leakage power for the higher-leakage technologies, we propose a new hardware-software technique where pipeline stalls are inserted into the processor after every instruction while maintaining the clock rate of the processor. The hardware units are designed to save leakage power while processing NOP instruction by putting the idle blocks into sleep mode. This technique is more effective when NOP cycle consumes less than 50% power than the regular instruction cycle. For the future work, power of the active cycles of the processor can be worked upon to further reduce the leakage and dynamic power. Also voltage reduction can be considered for further reduction in power when reducing the clock frequency, if the performance penalty can be met.

Bibliography

[1] International technology roadmap for semiconductors. http://public.itrs.net.

[2] http://www-device.eecs.berkeley.edu/ ptm:BSIM3files.

[3] A. Abdollahi, F. Fallah, and M. Pedram, "Leakage Current Reduction in CMOS VLSI Circuits by Input Vector Control," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 2, pp. 140–154, Feb. 2004.

[4] K. Agarwal, H. Deogun, D. Sylvester, and K. Nowka, "Power Gating with Multiple Sleep Modes," in *Proc. International Symposium on Quality Electronic Design*, 2006, pp. 633–637.

[5] V. D. Agrawal, "Low-power design by hazard filtering," in *Proc. 10th International Conf. on VLSI Design*, Jan. 1997, pp. 193–197.

[6] V. D. Agrawal, M. L. Bushnell, G. Parthasarathy, and R. Ramadoss, "Digital circuit design for minimum transient energy and a linear programming method," in *Proc. 12th International Conf. VLSI Design*, 1999, pp. 434–439.

[7] J. D. Alexander, "Simulation Based Power Estimation for Digital CMOS Technologies," Master's thesis, Auburn University, ECE Department, Dec. 2008.

[8] A. Arthurs and L. Ngo, "Analysis of the MIPS-32 Bit, Pipelined Processor using Synthesized VHDL." Department of Computer Science and Engineering, University of Arkansas.

[9] P. Ashenden, *The Designer's Guide to VHDL*. Morgan Kaufmann, 2002.

[10] S. Augsburger and B. Nikoli, "Combining Dual-Supply, Dual-Threshold and Transistor Sizing for Power Reduction," in *Proc. IEEE International Conference on Computer Design*, 2002, pp. 316–321.

[11] J. L. Ayala, A. Veidenbaum, and M. Lopez-Vallejo, "Power-Aware Compilation for Register File Energy Reduction," *International Journal of Parallel Programming*, vol. 31, no. 6, pp. 451–467, Dec 2003.

[12] G. Baccarani, M. Wordeman, and R. Dennard, "Generalized Scaling Theory and its Application to 1/4 Micrometer MOSFET Design," *IEEE Transactions on Electron Devices*, vol. ED-31, no. 4, pp. 452–462, April 1984.

[13] H. Bakoglu, *Circuits, Interconnections, and Packaging for VLSI*. Addison-Wesley, 1990.

[14] R. Bechade, R. Flaker, B. Kauffmann, S. Kenyon, C. London, S. Mahin, K. Nguyen, D. Pham, A. Roberts, S. Ventrone, and T. VonReyn, "A 32b 66MHz 1.8W Microprocessor," in *Proc. IEEE International Solid-State Circuits Conference*, Feb. 1994, pp. 208–209.

[15] A. Bellaur and M. I. Elmasry, *Low Power Digital CMOS Design: Circuits and Systems*, p. 90. Norwell, MA: Kluwer Publisher, 1996.

[16] A. Bellaur and M. I. Elmasry, *Low Power Digital CMOS Design: Circuits and Systems*, pp. 135–137. Norwell, MA: Kluwer Publisher, 1996.

[17] L. Benini, M. Favalli, and B. Ricco, "Analysis of Hazard Contributions to Power Dissipation in CMOS IC's," in *Proc. International Workshop on Low-Power Design*, Apr. 1994, pp. 27–32.

[18] L. Benini and G. D. Micheli, *Dynamic Power Management, Design Techniques and CAD Tools*. Springer, 1998.

[19] L. Benini, G. D. Micheli, E. Macii, M. Poncino, and S. Quer, "Reducing Power of Core Based Systems by Address Bus Encoding," *IEEE Trans. on VLSI Systems*, vol. 6, no. 4, pp. 554–562, Oct. 1998.

[20] D. Benson, Y. Bobra, and B. McWilliams, "Silicon Multichip Modules," in *International Workshop on Low-Power Design*, Aug. 1990.

[21] S. Borkar, "Design Challenges of Technology Scaling," *IEEE Micro*, vol. 19, no. 4, pp. 23–29, July 1999.

[22] T. D. Burd and R. W. Brodersen, *Energy Efficient Microprocessor Design*. Norwell, MA: Kluwer Academic Publisher, 2001.

[23] J. Burr and A. Peterson, "Energy Considerations in Multichip Module Based Multiprocessors," in *Proc. in International Conference on Computer Design*, April 1991, pp. 593–600.

[24] J. Burr and J. Shott, "A 200mV Self-Testing Encoder/Decoder using Stanford Ultra-Low-Power CMOS," in *Proc. of the International Solid State Circuits Conference*, 1994, pp. 84–85.

[25] B. H. Calhoun and A. P. Chandrakasan, "Standby Power Reduction Using Dynamic Voltage Scaling and Canary Flip-Flop Structures," *IEEE Journal of Solid-State Circuits*, vol. 39, no. 9, pp. 1504–1511, Sept. 2004.

[26] Y. Cao, T. Sato, D. Sylvester, M. Orshansky, and C. Hu, "New Paradigm of Predictive MOSFET and Interconnect Modeling for Early Circuit Design," in *Proc. of IEEE Custom Integrated Circuit Conference*, 2000, pp. 201–204.

[27] A. Chandrakasan, R. Allmon, A. Stratakos, and R. W. Brodersen, "Design of Portable Systems," in *Proc. Custom Integrated Circuits Conf.*, May 1994, pp. 259–266.

[28] A. Chandrakasan and R. W. Brodersen, "Minimizing Power Consumption in Digital CMOS Circuits," in *Proc. IEEE*, 1995, pp. 498–523.

[29] A. Chandrakasan, A. Burstein, and R. Brodersen, "A Low Power Chipset for Portable Multimedia Applications," in *Proc. International Solid-State Circuits Conference*, 1994, pp. 82–83.

[30] A. Chandrakasan, S. Sheng, and R. W. Brodersen, "Low-power CMOS digital design," *IEEE J. Solid State Circuits*, vol. 27, no. 4, pp. 473–484, April 1992.

[31] K. Chao and D. Wong, "Low Power Considerations in Floorplan Design," in *International Workshop on Low Power Design*, 1994, pp. 45–50.

[32] S. Chatre, C. D. Knutson, D. Margineantu, and C. Schulz-Key, "Improving DLX Performance by Taking Some of the Reduction Out of RISC," in *Proc. of the International Conference on Technical Informatics*, 1996.

[33] C. Chen and M. Sarrafzadeh, "Power Reduction by Simultaneous Voltage Scaling and Gate Sizing," in *Proc. of the Asia and South Pacific Design Automation Conference*, 2000, pp. 333–338.

[34] M. H. Chowdhury, J. Gjanci, and P. Khaled, "Innovative Power Gating for Leakage Reduction," in *Proc. International Symposium on Circuits and Systems*, 2008, pp. 1568–1571.

[35] R. H. Dennard, F. H. Gaensslen, H. N. Yu, V. L. Rideout, E. Bassous, and A. R. LeBlanc, "Design of Ion-Implanted MOSFETs with Very Small Physical Dimensions," *IEEE Journal of Solid State Circuits*, vol. SC-9, no. 5, pp. 256–258, Oct. 1974.

[36] D. Dobberpuhl, R. Witek, R. Allmon, R. Anglin, S. Britton, L. Chao, R. Conrad, D. Dever, B. Gieseke, G. Hoeppner, J. Kowaleski, K. Kuchler, M. Ladd, M. Leary, L. Madden, E. McLellan, D. Meyer, J. Montanaro, D. Priore, V. Rajagopalan, S. Samudrala, and S. Santhanam, "A 200MHz 64b Dual-Issue CMOS Microprocessor," in *Proc. 39th IEEE International Solid-State Circuits Conference*, Feb. 1992, pp. 106–107.

[37] F. Emnett and M. Biegel, "Power Reduction Through RTL Clock Gating." Presesnted at Synopsys Users Group (SNUG), 2000.

[38] D. Ernst, N. S. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge, "Razor: A Low-Power Pipeline Based on Circuit-Level Timing Speculation," in *Proc. 36th Annual IEEE/ACM International Symp. on Microarchitecture*, Dec. 2003, pp. 7–18.

[39] M. Farrahi, G. E. Tellez, and M. Sarrafzadeh, "Memory segmentation to exploit sleep mode operation," in *Proc. Design Automation Conference*, 1995, pp. 36–41.

[40] K. Flautner, N. S. Kim, S. Martin, D. Blaauw, and T. Mudge, "Drowsy Caches: Simple Techniques for Reducing Leakage Power," in *Proc. International Symposium on Computer Architecture*, 2002, pp. 148–157.

[41] R. Gonzalez, B. M. Gordon, and M. A. Horowitz, "Supply and Threshold Voltage Scaling for Low Power CMOS," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 8, pp. 1210–1216, Aug. 1997.

[42] F. Hamzaoglu and M. R. Stan, "Circuit-Level Techniques to Control Gate Leakage for sub-100nm CMOS," in *International Symposium on Low Power Electronics and Design*, 2002, pp. 60–63.

[43] M. Hans, "Architectural Aspects of Design for Low Static Power Consumption," Master's thesis, Computer Science and Engineering Division, Technical University of Denmark, 2004.

[44] J. G. Hansen, "Design of CMOS Cell Libraries for Minimal Leakage Currents," Master's thesis, Dept. of Informatics and Mathematical Modeling, Technical University of Denmark, 2004.

[45] Z. L. He, K. K. Chan, C. Y. Tsui, and M. L. Liou, "Low Power Motion Estimation Design using Adaptive Pixel Truncation," in *Proc. International Symp. on Low Power Electronics and Design*, 1997, pp. 167–172.

[46] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann, 1990.

[47] S. Hines, J. Green, G. Tyson, and D. Whalley, "Improving Program Efficiency by Packing Instructions into Registers," in *Proc. of the 2005 ACM/IEEE International Symposium on Computer Architecture*, 2005, pp. 260–271.

[48] S. Hines, G. Tyson, and D. Whalley, "Improving the Energy and Execution Efficiency of a Small Instruction Cache by using an Instruction Register File," in *Proc. of the Watson Conference on Interaction between Architecture, Circuits, and Compilers*, 2005, pp. 160–169.

[49] F. Hu, *Process-Variation-Resistant Dynamic Power Optimization for VLSI Circuits*. PhD thesis, Auburn University, ECE Department, May 2006.

[50] F. Hu and V. D. Agrawal, "Dual-Transition Glitch Filtering in Probabilistic Waveform Power Estimation," in *Proc. IEEE Great Lakes Symp. on VLSI*, Apr. 2005, pp. 357–360.

[51] Z. Hu, A. Buyuktosunoglu, V. Srinivasan, V. Zyuban, H. Jacobson, and P. Bose, "Microarchitectural Techniques for Power Gating of Execution Units," in *International Symposium on Low Power Electronics and Design*, 2004, pp. 32–37.

[52] A. Iyer and D. Marculescu, "Power Efficiency of Voltage Scaling in Multiple Clock, Multiple Voltage Cores," in *Proc. IEEE/ACM International Conference on Computer Aided Design*, 2002, pp. 379–386.

[53] P. Kalla, X. S. Hu, and J. Henkel, "Distance-Based Recent Use (DRU): An Enhancement to Instruction Cache Replacement Policies for Transition Energy Reduction," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 14, no. 1, pp. 69–80, Jan. 2006.

[54] J. T. Kao and A. P. Chandrakasan, "Dual-Threshold Voltage Techniques for Low-Power Digital Circuits," *IEEE Journal of Solid-State Circuits*, vol. 35, no. 7, pp. 1009–1018, July 2000.

[55] N. S. Kim, K. Flautner, D. Blaauw, and T. Mudge, "Circuit and Microarchitectural Techniques for Reducing Cache Leakage Power," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 2, pp. 167–184, Feb. 2004.

[56] P. E. Landman, *Low-Power Architectural Design Methodologies*. PhD thesis, University of California, Berkeley, 1994.

[57] C. Lee, J. K. Lee, T. Hwang, and S. Tsai, "Compiler Optimization on Instruction Scheduling for Low Power," in *Proc. of the 13th International Symposium on System Synthesis*, 2000, pp. 55–60.

[58] C. E. Leiserson, *Area-Efficient VLSI Computation*. PhD thesis, Massachusetts Institute of Technology, 1983.

[59] C. E. Leiserson, F. M. Rose, and J. B. Saxe, "Optimizing Synchronous Circuitry by Retiming," in *Proc. Third Caltech Conf.*, 1983, pp. 86–116.

[60] C. E. Leiserson and J. B. Saxe, "Optimizing Synchronous Systems," *Journal of VLSI and Computer Systems*, vol. 1, no. 1, pp. 41–67, 1983.

[61] C. E. Leiserson and J. B. Saxe, "Retiming Synchronous Circuitry," in *Algorithmica*, 1991, pp. 5–35.

[62] H. Li, S. Bhunia, Y. Chen, T. N. Vijaykumar, and K. Roy, "Deterministic Clock Gating for Microprocessor Power Reduction," in *Proc. 9th International Symposium on High-Performance Computer Architecture*, Feb. 2003, pp. 113–122.

[63] D. Liu and C. Svensson, "Trading Speed for Low Power by Choice of Supply and Threshold Voltages," *IEEE J. Solid-State Circuits*, pp. 10–17, Jan. 1993.

[64] W. Liu, "Techniques of Leakage Power Reduction in Nanoscale Circuits: A Survey." IMM Report, Dept. of Informatics and Mathematical Modeling, Technical University of Denmark, 2007.

[65] P. Lotfi-Kamran, A. Rahmani, A. Salehpour, A. Afzali-Kusha, and Z. Navabi, "Stall Power Reduction in Pipelined Architecture Processors," in *Proc. of 21st International Conference on VLSI Design*, 2008, pp. 541–546.

[66] Y. Lu, *Power and Performance Optimization of Static CMOS Circuits with Process Variation.* PhD thesis, Auburn University, ECE Department, Aug. 2007.

[67] Y. Lu and V. D. Agrawal, "Leakage and Dynamic Glitch Power Minimization Using Integer Linear Programming for Vth Assignment and Path Balancing," in *Proc. International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, 2005, pp. 217–226.

[68] Y. Lu and V. D. Agrawal, "CMOS Leakage and Glitch Minimization for Power-Performance Tradeoff," *Journal of Low Power Electronics*, vol. 2, no. 3, pp. 378–387, Dec. 2006.

[69] Y. Lu and V. D. Agrawal, "Total Power Minimization in Glitch-Free CMOS Circuits Considering Process Variation," in *Proc. of the 21st International Conference on VLSI Design*, Jan. 2008, pp. 527–532.

[70] S. M. Martin, K. Flautner, T. Mudge, and D. Blaauw, "Combined Dynamic Voltage Scaling and Adaptive Body Biasing for Lower Power Microprocessors under Dynamic Workloads," in *Proc. IEEE/ACM International Conference on Computer Aided Design*, 2002, pp. 721–725.

[71] B. Mathew, A. Davis, and M. Parker, "A Low Power Architecture for Embedded Perception," in *Proc. of International Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES)*, 2004, pp. 46–56.

[72] Y. Meng, T. Sherwood, and R. Kastner, "On the Limits of Leakage Power Reduction in Caches," in *International Symposium on High-Performance Computer Architecture*, 2005, pp. 154–165.

[73] B. Moyer, "Low Power Design for Embedded Processors," in *Proc. of IEEE*, 2001, pp. 1576–1587.

[74] R. S. Muller and T. I. Kamins, *Device Electronics for Integrated Circuits.* John Wiley and Sons, 1986.

[75] R. Murgai, R. Brayton, and A. Sangiovanni-Vincentelli, "Decomposition of Logic Functions for Minimum Transition Activity," in *Proc. International Workshop on Low-Power Design*, Apr. 1994, pp. 33–38.

[76] C. Nagendra, U. Mehta, R. Owens, and M. Irwin, "A Comparison of the Power-Delay Characteristics of CMOS Adders," in *Proc. International Workshop on Low Power Design*, Apr. 1994, pp. 231–236.

[77] K. Najeeb, V. V. R. Konda, S. S. Hari, V. Kamakoti, and V. M. Vedula, "Power Virus Generation Using Behavioral Models of Circuits," in *Proc. 25th IEEE VLSI Test Symposium*, 2007, pp. 35–40.

[78] S. R. Nassif, H. Jiang, and M. Marek-Sadowska, "Benefits and Costs of Power-Gating Technique," in *International Conference on Computer Design*, 2005, pp. 559–566.

[79] K. Natarajan, H. Hanson, S. W. Keckler, C. R. Moore, and D. Burger, "Microprocessor Pipeline Energy Analysis," in *Proc. of the 2003 International Symposium on Low power Electronics and Design*, 2003, pp. 282–287.

[80] D. A. Patterson and J. L. Hennessy, *Computer Organization and Design: The Hardware/Software Interface, Fourth Edition*. Morgan Kaufmann, 2009.

[81] T. Pering, T. Burd, and R. Brodersen, "Dynamic Voltage Scaling and the Design of a Low-Power Microprocessor System," in *Proc. Power-Driven Microarchitecture Workshop*, 1998, pp. 251–259.

[82] T. S. Perry, "Gordon Moore's Next Act," *IEEE Spectrum*, vol. 45, no. 5, May 2008.

[83] D. Pham, M. Alexander, A. Arizpe, B. Burgess, C. Dietz, L. Eisen, R. El-Kareh, J. Eno, S. Gary, G. Gerosa, B. Goins, J. Golab, R. Golla, R. Harris, B. Ho, Y.-W. Ho, K. Hoover, C. Hunter, P. Ippolito, R. Jessani, J. Kahle, K. R. Kishore, B. Kuttanna, S. Litch, S. Mallick, T. Ngo, D. Ogden, C. Olson, S.-H. Park, R. Patel, M. Pham, J. Prado, S. Reeve, R. Reininger, H. Sanchez, M. Schiffli, J. Slaton, G. Thuraisingham, K. Torku, C. Tran, N. Vanderschaaf, and P. Voldstad, "A 3.0W 75SPECint92 85SPECfp92 Superscalar RISC Microprocessor," in *Proc. International Solid-State Circuits Conference*, Feb. 1994, pp. 212–213.

[84] J. Rabaey, *Digital Integrated Circuits: A Design Perspective*. Prentice Hall, 1995.

[85] H. Rahman and C. Chakrabarti, "An Efficient Control Point Insertion Technique for Leakage Reduction of Scaled CMOS Circuits," *IEEE Transactions on Circuits and Systems-II: Express Briefs*, vol. 52, no. 8, pp. 496–500, Aug. 2005.

[86] T. Raja, "A Reduced Constraint Set Linear Program for Low-Power Design of Digital Circuit," Master's thesis, Rutgers University, ECE Department, Mar. 2002.

[87] T. Raja, *Minimum Dynamic Power CMOS Design with Variable Input Delay Logic*. PhD thesis, Rutgers University, ECE Department, May 2004.

[88] T. Raja, V. D. Agrawal, and M. L. Bushnell, "Minimum Dynamic Power CMOS Circuit Design by a Reduced Constraint Set Linear Program," in *Proc. 16th International Conf. VLSI Design*, Jan. 2003, pp. 527–532.

[89] T. Raja, V. D. Agrawal, and M. L. Bushnell, "'CMOS Circuit Design for Minimum Dynamic Power and Highest Speed," in *Proc. 17th International Conf. VLSI Design*, Jan. 2004, pp. 1035–1040.

[90] T. Raja, V. D. Agrawal, and M. L. Bushnell, "Transistor Sizing of Logic Gates to Maximize Input Delay Variability," *Journal of Low Power Electronics*, vol. 2, no. 1, pp. 121–128, Apr. 2006.

[91] J. Sacha and M. J. Irwin, "Number Representations for Predicting Data bus Power Dissipation," in *Proc. of International Conference of Acoustics Speech and Signal Processing*, 1998, pp. 163–168.

[92] R. Saleh, "Power and Low-Power Design." Dept. of ECE, University of British Columbia.

[93] M. J. Schulte, J. E. Stine, and J. G. Jansen, "Reduced Power Dissipation Through Truncated Multiplication," in *Proc. IEEE Alessandro Volta Memorial Workshop Low-Power Design*, March 1999, pp. 61–69.

[94] J. Schutz, "A 3.3V 0.6μm BiCMOS Superscalar Microprocessor," in *Proc. IEEE International Solid-State Circuits Conference*, Feb. 1994, pp. 202–203.

[95] D. Snowdon, S. Ruocco, and G. Heiser, "Power Management and Dynamic Voltage Scaling: Myths and Facts," in *Proc. of the 2005 Workshop on Power Aware Real-time Computing*, 2005.

[96] R. Srinivasan, J. Cook, and L. Eisen, "Evaluating Instruction Reorderings and Transformations for Microarchitecture Power Reduction," in *Proc. of the 6th Annual Austin CAS International Conference*, 2005.

[97] S. Steinke, R. Schwarz, L. Wehmeyer, and P. Marwedel, "Low Power Code Generation of a RISC Processor by Register Pipelining." Technical Report 754, University of Dortmund, Dept. of Computer Science XII, 2001.

[98] A. Stratakos, R. W. Brodersen, and S. R. Sanders, "High-Efficiency Low-Voltage DC-DC Conversion for Portable Applications," in *International Workshop on Low-Power Design*, April 1994, pp. 105–110.

[99] C. Su, C. Tsui, and A. Despain, "Low Power Architecture Design and Compilation Techniques for High-Performance Processors," in *Proc. of IEEE COMPCON*, 1994, pp. 489–498.

[100] S. W. Sun and P. Tsui, "Limitation of CMOS Supply-Voltage Scaling by MOSFET Threshold Variation," in *Proc. IEEE Custom Integrated Circuits Conf.*, 1994, pp. 43–48.

[101] V. Sundararajan and K. K. Parhi, "Low Power Synthesis of Dual Threshold Voltage CMOS VLSI Circuits," in *Proc. International Symp. on Low Power Electronics and Design*, 1999, pp. 139–144.

[102] V. Tiwari, P. Ashar, and S. Mailk, "Technology Mapping for Low Power," in *Proc. 30th Design Automation Conference*, June 1993, pp. 74–79.

[103] Y. F. Tong, R. A. Rutenbar, and D. F. Nagle, "Minimizing Floating-Point Power Dissipation via Bit-Width Reduction," in *Power Driven Microarchitecture Workshop in Conjunction with ISCA'98*, June 1998, pp. 114–118.

[104] C. Tsui, M. Pedram, and A. Despain, "Technology Decomposition and Mapping Targeting Low Power Dissipation," in *Proc. 30th Design Automation Conference*, June 1993, pp. 68–73.

[105] S. Uppalapati, "Low Power Design of Standard Cell Digital VLSI Circuits," Master's thesis, Rutgers University, ECE Department, May 2004.

[106] S. Uppalapati, M. L. Bushnell, and V. D. Agrawal, "Glitch-Free Design of Low Power ASICS using Customized Resistive Feedthrough Cells," in *Proc. of the 9th VLSI Design and Test Symposium*, Aug. 2005, pp. 41–48.

[107] K. Usami and M. Horowitz, "Cluster Voltage Scaling Technique for Low Power Design," in *International Symposium on Low Power Design*, 1995, pp. 3–8.

[108] H. Vaishnav and M. Pedram, "PCUBE: A Performance Driven Placement Algorithm for Lower Power Designs," in *Proc. of the Euro-DAC*, 1993, pp. 72–77.

[109] H. J. M. Veendrick, "Short-Circuit Dissipation of Static CMOS Circuitry and its Impact on the Design of Buffer Circuits," *IEEE J. Solid-State Circuits*, pp. 468–473, Aug. 1984.

[110] S. R. Vemuru and N. Scheinberg, "Short Circuit Power Dissipation Estimation for CMOS Logic Gates," *IEEE Trans. Circuits Syst.*, pp. 762–765, Nov. 1994.

[111] F. M. Wanlass and C. T. Sah, "Nanowatt Logic Using Field-Effect Metal-Oxide Semiconductor Triodes," in *Proc. International Solid State Circuits Conference*, Feb. 1963, pp. 32–33.

[112] N. H. E. Weste and D. Harris, *CMOS VLSI Design, A Circuits and Systems Perspective, 3rd Edition.* Addison Wesley, 2004.

[113] W. Ye, N. Vijaykrishnan, M. T. Kandemir, and M. J. Irwin, "The Design and Use of Simple Power: a Cycle-Accurate Energy Estimation Tool," in *Proc. of Design Automation Conference*, 2000, pp. 340–345.

[114] N. K. Yeung, Y.-H. Sutu, T. Y.-F. Su, E. T. Pak, C.-C. Chao, S. Akki, D. D. Yau, and R. Lodenquai, "The Design of a 55SPECint92 RISC Processor under 2W," in *Proc. IEEE International Solid-State Circuits Conference*, Feb. 1994, pp. 206–207.

[115] B. Yu, *A New Dynamic Power Cut-Off Technology (DPCT) for Leakage Reduction in Deep Submicron VLSI CMOS Circuits*. PhD thesis, Rutgers University, ECE Department, Oct. 2007.

[116] B. Yu and M. L. Bushnell, "A Novel Dynamic Power Cut-off Technique (DPCT) for Active Leakage Reduction in Deep Submicron CMOS Circuits," in *Proc. International Symp. on Low Power Electronics and Design*, 2006, pp. 214–219.