A Study of Leakage Power Reduction Mechanisms on Functional Units and TLBs for Embedded Processors

Zhao LEI

A dissertation submitted in partial fulfillment of
the requirements for the degree of

DOCTOR OF PHILOSOPHY

School of Science for Open and Environmental Systems
Graduate School of Science and Technology
Keio University

2011

Preface

Power consumption has been widely recognized as a first-class design constraint for embedded processors, due to its impact on operation reliability, system density, and integration costs. While dynamic power has represented the predominant factor in CMOS circuits for many years, the leakage power, which is consumed by each transistor even when no active switching is taking place, is increasingly prominent with technology scale. Now, suppressing the leakage power of embedded processors, especially for battery-driven devices, is a critical challenge facing the embedded system community.

Leakage-efficient design requires an in-depth examination of each system component. In this thesis, we explore the leakage reduction mechanisms on functional units, instruction TLB, and data TLB, all of which take up a significant share of the total leakage consumption of embedded processors but have not been well studied due to their high access frequency and utilization.

As for functional units, we propose a framework to Power Gating (PG) each of the units at runtime by integrating circuit-level, architecture-level, and system software techniques. At circuit-level, we propose a fine-grained power gating technique, which has nano-second order wakeup latency and can be implemented at arbitrary granularity. At architecture-level, a PG control scheme, which keeps a functional unit active only when being used, has been applied. In addition, BET-aware (Break Even Time) PG control schemes, which are guided by the system software (compiler and operating system), have also been proposed to achieve maximum leakage reduction effects.

As for the instruction TLB (iTLB), we exploit the spatial locality of the page-based iTLB references. By inserting a small size storage component, which keeps the recent address-translation information, between the processor and the iTLB, a majority of address-translation requests can be satisfied with the small component without accessing the iTLB. Then, with integration of the Dual Voltage Supply (DVS) technique, the iTLB can be put into low-leakage mode (with the lower voltage supply) and restored to the active mode only when the iTLB look-up becomes necessary. Based on such a design philosophy, three different leakage control policies have also been proposed to achieve the best leakage reduction efficiency.

As for the data TLB (dTLB), we exploit the temporary locality dTLB references. By dividing the overall execution time into smaller time slices, we can observe the dTLB referencing in a finer time resolution, and the locality of dTLB references in and between adjacent slices can be utilized to recognize the contributive dTLB entries in each slice. Then, with integration of the DVS technique,

those non-contributive entries can be put into low leakage mode dynamically.

The proposed mechanisms are evaluated in terms of leakage power consumption by using real-chip/post-layout evaluation based on 65nm CMOS technology. Evaluation results show that the proposed mechanisms can reduce the leakage power consumption of the functional units, iTLB and dTLB by 20%, 50%, and 37% respectively.

Acknowledgments

My doctoral study has been supported by a number of people. While I would like to acknowledge individually each by name, the short list below is likely incomplete. I apologize in advance for such omission, and convey my deepened respect and admiration to all who have encouraged and supported me in the long road to my dissertation.

First and foremost, I would like to express my deepest gratitude to my supervisor, Professor Hideharu Amano, for his supporting and guidance throughout my study at Keio University. He created an environment that allowed me to get comfortable, grow, and learn; and I could always count on him when I was stuck on issues, both in research and in daily life. Without him, this dissertation would not have be possible.

I am grateful to my doctoral committee members, Professor Iwao Sasase, Associate Professor Kenji Kono, and Associate Professor Nobuyuki Yamasaki for their careful reviews and valuable comments.

It has been a wonderful experience to participate in the Geyser group and collaborate with researchers from other universities. I would like to thank Prof. Mitaro Namiki at Tokyo University of Agriculture and Industry, Prof. Kimiyoshi Usami at Shibaura Institute of Technology, Prof. Hiroshi Nakamura at University of Tokyo, and Associate Prof. Masaaki Kondo at University of Electro-Communications. Their creative vision and constructive ideas have been crucial to the success of the whole project and my research. Thanks also goes to all other Geyser members, especially Naomi Seki and Daisuke Ikebuchi.

I would like to thank all “hlab” members for providing me daily inspirations and so much fun. Studying abroad can be tough, but you have made it much easier.

A special thank is owed to MEXT scholarship for giving me a great opportunity to pursue my Ph.D. Degree in Keio University. I also want to thank Amano Institute of Technology for providing financial support during the last year of my course.

Finally, I am grateful to my parents and my lovely wife. Thanks so much for all of the love and sacrifice that you have done for me.

Zhao LEI

Yokohama, Japan

April 2011

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

The increasing prominence of portable electronics and consumer-oriented devices has become a fundamental driving factor in the design of high-performance low-power embedded processors. In a large part, the interminable requirements on performance have been satisfied by the exponential increases in device density and operating frequency through VLSI technology scaling [1] – with each new technology generation, there is a 2~3× performance increase for embedded processors. This, however, has led to exponential increases in power consumption [2]. High power consumption introduces challenges to various aspects of embedded systems. It increases the cost of cooling and packaging design, reduces system reliability, complicates power supply circuitry design, and reduces battery lifetime. Now, power consumption has been widely recognized as a first-class design constraint for embedded processors [3].

Until very recently, the primary source of power consumption in digital CMOS circuits has been dynamic power. Dynamic power arises from the dynamic switching of load capacitances, and Moore's law [4] has helped to control it – the continuing reduction in feature size reduces capacitance and the accompanying reductions in supply voltage help to reduce the dynamic switching energy per operation. However, to maintain performance scaling, threshold voltages must also be scaled along with supply voltage. Lowering threshold voltage increases leakage current exponentially, and within a few process generations it is predicted power consumption from static leakage current could be comparable to dynamic power [5].

Fig.1.1 compares the leakage power consumption and its dynamic counterpart with data from existing technologies and projections based on the international technology roadmap for semiconductor (ITRS) [6]. As it can be seen, even in current-generation technology, sub-threshold leakage power consumption is comparable to the dynamic power consumption, and the fraction of the leakage power will increase significantly in the near future[1]. In fact, the off-state sub-threshold leakage

[1]The techniques presented in this thesis address sub-threshold leakage, which is the principal component of static power consumption. As will be shown in next chapter, another type of leakage, called gate oxide leakage, is not addressed

component of the total power in a processor may exceed dynamic power as the technology decreases below the 65 nm technology node, according to a projection from Intel [7].

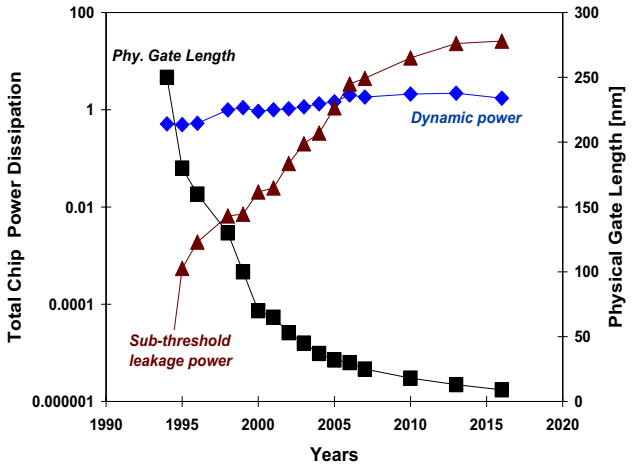


Figure 1.1: Total Chip Dynamic and Leakage Power Dissipation Trends

Moreover, the trend toward ever more complex embedded processors further exacerbates the situation, as large numbers of transistors are added for relatively small improvements in performance. These additional transistors may dissipate considerable leakage power even when not actively switching. Currently, suppressing the leakage power of embedded processors, especially for battery-driven devices, is a critical challenge facing the embedded system community.

1.2 Objective

Leakage-efficient design requires an in-depth examination of each system component and has become a very active research field in last decade. Fig.1.2 presents a high-level block diagram of a embedded processor. The diagram is based on MIPS R3000 architecture [8] which includes all essential function blocks of modern embedded processors – pipeline structure, register files (RF), functional units (FUs), level-1 instruction cache (iCache) and data cache (dCache), memory controller (Mem Cont.), instruction translation lookaside buffer (iTLB), and data translation lookaside buffer (dTLB). Previous leakage reduction mechanisms were mainly applied to the on-chip caches, because the large semiconductor footprints and regular access properties of cache memory made them an excellent target for developing high-level mechanisms to fight leakage. With integration of circuit-level low-leakage techniques, those mechanisms can reduce the leakage power by appropriately switching the whole or parts of the cache memory between the normal mode and the low-leakage mode. Although significant leakage reduction effects on cache memory have been achieved, leakage reduction mechanisms on other components are also indispensable to guarantee the overall leakage reduction

architecturally but rather at the process level.

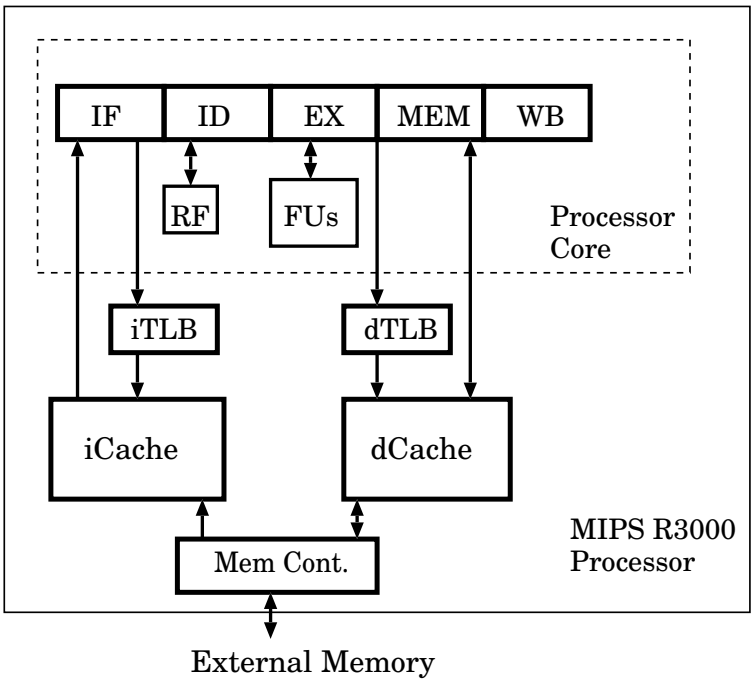


Figure 1.2: Block Diagram of a MIPS R3000 Processor

effects of a processor.

In this thesis, we explore the leakage reduction mechanisms on other on-chip components. Based on the leakage analysis of each on-chip component of a representative embedded processor – MIPS R3000, we select three of them as the leakage-reduction target – functional units, instruction TLB, and data TLB – all of which take up a significant share of the total leakage consumption of embedded processors but have not been well studied due to their high access frequency and utilization.

1.3 Contribution

In this thesis, we select three different on-chip components – functional units, instruction TLB, and data TLB as the leakage reduction target based on a comprehensive leakage analysis of a MIPS R3000 processor.

- As for functional units, we propose a framework to Power Gating (PG) each of the units at run-time by integrating circuit-level, architecture-level, and system software techniques.

  1) At circuit-level, we propose a fine-grained power gating technique, which has nano-second order wakeup latency and can be implemented at arbitrary granularity.

  2) At architecture-level, a PG control scheme, which keeps a functional unit active only when being used, has been applied.

  3) In addition, BET-aware (Break Even Time) PG control schemes, which are guided by the

system software (compiler and operating system), have also been proposed to achieve maximum leakage reduction effects.

- As for the instruction TLB (iTLB), we propose a leakage-efficient structure by exploiting the spatial locality of the page-based iTLB references. Following topics have been discussed:

  1) We explore the leakage reduction opportunity embodied in the instruction stream; that is, when a program enters a physical page, following instructions tend to be fetched from the same page for a considerable long time.

  2) We insert a small size storage component, which keeps the recent address-translation information, between the processor and the iTLB. Thus, a majority of address-translation requests can be satisfied with the small component without accessing the iTLB.

  3) We implement the iTLB with integration of the Dual Voltage Supply (DVS) technique. The iTLB can be put into low-leakage mode (with the lower voltage supply) and restored to the active mode only when the iTLB look-up becomes necessary.

  4) Based on such a design philosophy, three different leakage control policies have also been proposed to achieve the best leakage reduction efficiency.

- As for the data TLB (dTLB), we propose a leakage-efficient structure by exploiting the temporary locality dTLB references. Following topics have been discussed:

  1) We explore the leakage reduction opportunity embodied in data references; that is, in a short time period, only a small subset of dTLB entries actually serves for the data-address translation requests.

  2) By dividing the overall execution time into smaller time slices, we can observe the dTLB referencing in a finer time resolution, and the locality of dTLB references in and between adjacent slices can be utilized. A mechanism has been proposed to recognize the only the contributive dTLB entries in each time slice.

  3) We implement the dTLB with integration of the DVS technique, those non-contributive entries in each time slot can be put into low leakage mode dynamically.

The proposed mechanisms are evaluated in terms of leakage power consumption by using real-chip/post-layout evaluation based on 65nm CMOS technology. Evaluation results show that the proposed mechanisms can reduce the leakage power consumption of the functional units, iTLB and dTLB by 20%, 50%, and 37% respectively.

1.4 Thesis Organization

This thesis is organized as follows, and the thesis organization is illustrated in Fig.1.3. Chapter 2 introduces the underlying mechanics of leakage. This chapter also surveys current circuit and ar-

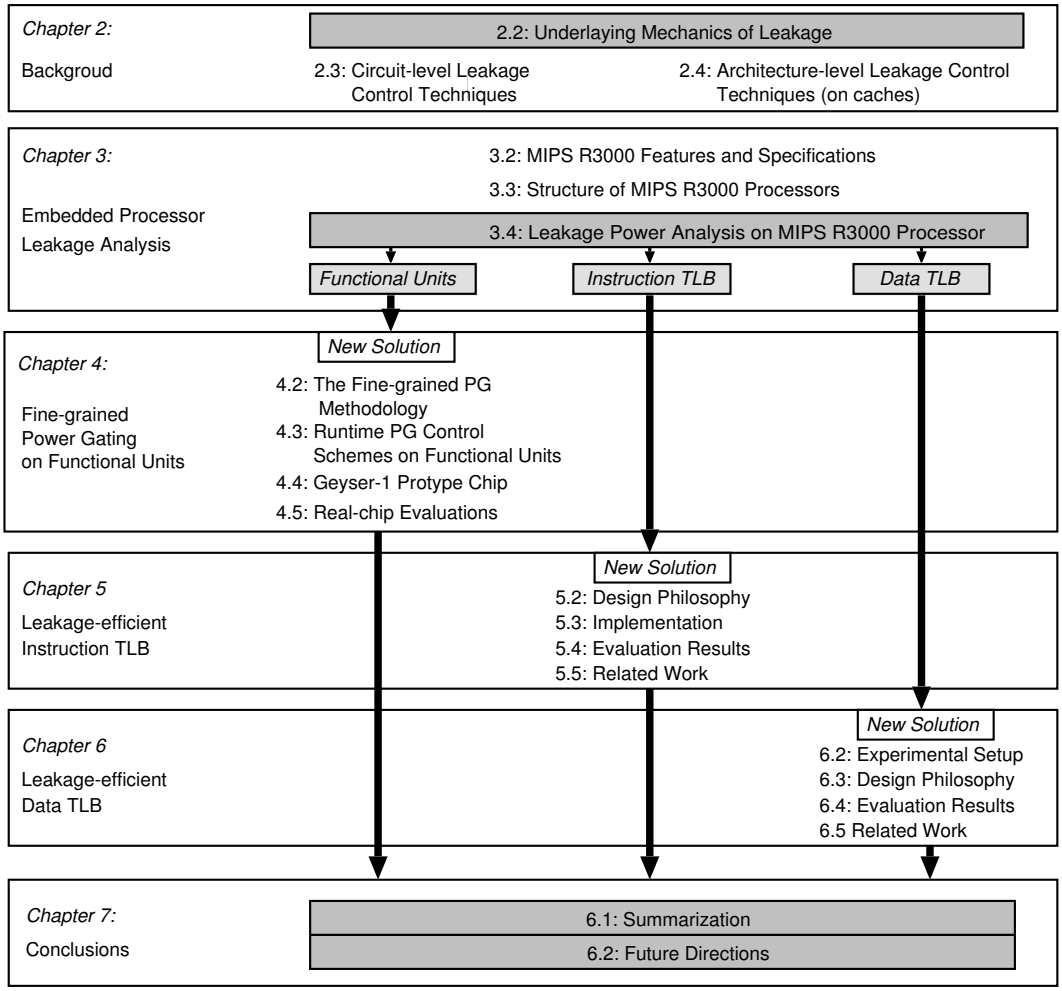


Figure 1.3: Thesis organization

chitectural technologies on microprocessors. Since leakage-efficient design requires an in-depth examination of each system component, Chapter 3 presents the leakage consumption of each on-chip component of a representative embedded processor – MIPS R3000, and based on the analysis results, three on-chip components – functional units, instruction TLB, and data TLB are picked up the leakage-reduction targets of this thesis. Chapter 4, 5, and 6 explore the leakage reduction mechanisms on functional units, iTLB, and dTLB respectively. Chapter 7 summarizes the proposed mechanisms and provides a discussion about the future research directions. Then it concludes this thesis.

Chapter 2

Background

2.1 Introduction

Static power consumption has grown to a significant portion of total power consumption in recent years. In CMOS technology, static power consumption is due to the imperfect nature of transistors which "leak" current – thereby constantly consuming power – even when they are not switching. The advent of this form of static power, called leakage power, was forecasted early on [1] [5], giving architects the opportunity to propose techniques to address it. Such techniques are the focus of this chapter.

Considerable work to reduce leakage power consumption is taking place at the process level [9]. In fact, process solutions such as the high-k dielectric materials in Intel's 45 nm process technology, are already employed. Addressing the problem at the circuit level and architectural level are, however, indispensable because they can be used orthogonally to process technology solutions. The importance of circuit and architectural techniques is magnified by the exponential dependence of leakage power to various operating parameters such as supply voltage (Vdd), temperature (T), and threshold voltage (Vth). Exponential dependence implies that a leakage reduction solution that works well at some specific operating conditions may not be enough – the problem is bound to reappear with the same intensity as before but at higher temperatures or lower voltages.

In this chapter we briefly illustrate the circuit-level leakage reduction techniques after a short introduction on underlying mechanics of leakage. It is important to note that the techniques presented in this chapter address a specific type of leakage, called sub-threshold leakage. Another type of leakage, called gate oxide leakage, is not addressed at the circuit level but rather at the process level.

2.2 Leakage Power

Static power is so called because it is consumed by every transistor even when no active switching is taking place. In older technologies (e.g., NMOS, TTL, ECL, etc.) it is an inherent problem, because a path from Vdd to ground is open even when transistors are not switching. With the advent of CMOS, static power became less of a concern because the complementary gate design prevents open paths from Vdd to ground. Unfortunately, static power resurfaced in CMOS in the form of leakage power. In the latest process generations leakage power increases exponentially, principally because of reductions in the threshold voltage. Leakage power increased to levels never seen before in CMOS – levels comparable to the dynamic (switching) power consumption – when technology scaling entered the deep-submicron territory in feature size (<180 nm). Currently, 20% ~ 40% of the total power consumption is attributed to leakage power. [10]

CMOS leakage power arises due to leakage currents. The total leakage current (I_{leak}) times the supply voltage gives the leakage power consumption, P_{leak}:

$$P_{leak} = V \times I_{leak}, \tag{2.1}$$

Leakage currents are a manifestation of the true analog nature of transistors, as opposed to our idealized view of them as perfect digital switches. The state of a transistor (on or off) is controlled by the voltage on its gate terminal. If this voltage is above the threshold voltage (Vth) the channel beneath the gate conducts, allowing current in the on state (Ion) to flow from the source (Vdd) to the drain (GND, ground). In the opposite case (gate voltage below Vth), we like to think that the transistor is off (perfect insulator). But in reality transistors leak – leakage currents flow even in their off state. As shown in Fig.2.1, this is evident in the I-V curve where current flows even below the threshold voltage where the device is supposed to be “off”.

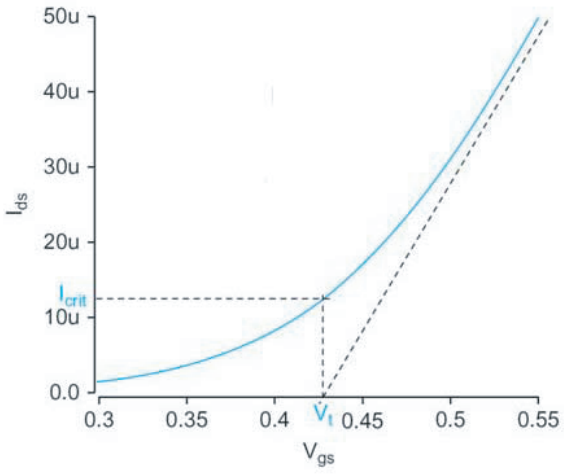


Figure 2.1: I-V Curve for a nMOS

The current that flows from source to drain when the transistor is off is called sub-threshold

leakage. But that is not all. There are five more types of leakage [10]: reverse-biased-junction leakage, gate-induced-drain leakage, gate-oxide leakage, gate-current leakage, and punch through leakage. The sub-threshold leakage and gate-oxide leakage dominate the total leakage current in devices. Both increase exponentially with each new technology generation with the gate-oxide leakage significantly outpacing the sub-threshold leakage.

In sub-micron technologies, sub-threshold and gate leakage are the cost we have to pay for the increased speed afforded by scaling. Supply voltage scaling attempts to curb an increase in dynamic power. Unfortunately, this strategy also leads to an enormous increase in the sub-threshold and gate leakage problem. This explains why static power has been gaining on dynamic power as a percentage of the total power consumption with every process generation.

2.2.1 Sub-threshold Leakage

Sub-threshold leakage increases with technology scaling due to Vdd scaling. The supply voltage (Vdd) is scaled along with other physical quantities to reduce dynamic power consumption. Scaling solely the supply voltage, however, increases the delay (switching speed) of the transistor. This is because the delay is proportional to the inverse of the current that flows in the on state – the Ion current.

$$Delay \propto \frac{1}{Ion} \propto \frac{Vdd}{(Vdd - Vth)^{\alpha}}, \tag{2.2}$$

This current, Ion, is a function of the supply voltage and the difference between the supply voltage and the threshold voltage (Vth). The factor α is a technology-dependent factor taking values greater than 1 (between 1.2 and 1.6 for recent technologies) [11]. Since Vdd is lowered in order to maintain the speed increase from scaling, the only course of action is to also lower the threshold voltage. Herein lies the problem: sub-threshold leakage increases exponentially with lower threshold voltage.

To understand the basic mechanisms for leakage reduction, we have to take a closer look at the formulas describing leakage current. We base our discussion on the Berkeley Predictive Model (BSIM3V3.2) formula for sub-threshold leakage [12] (which is also the starting point for the simplified Butts and Sohi models [13]). The formula describing the sub-threshold leakage current, I_{Dsub}, is:

$$I_{Dsub} = I_{s0}(1 - e^{\frac{-Vds}{\nu_t}})e^{\frac{Vgs-Vth-Voff}{n\nu_t}}, \tag{2.3}$$

Where

$$I_{s0} = K(W/L){\nu_t}^2, \tag{2.4}$$

$$V_{th} = V_{th0} - \gamma V_{bs} - \eta V_{ds}, \tag{2.5}$$

and Vds is the voltage bias across the drain and the source, and Vgs is the voltage bias across the gate

and source terminal. $Voff$ is an empirically determined BSIM model parameter and ν_t ($\nu_t = kT/q$) is a physical parameter called thermal voltage1 which is proportional to the temperature, T. γ is the linearized body effect coefficient; V_{bs} is the source to body voltage; η is the DIBL (drain induced barrier lowering) coefficient; and K is a process constant. The term n encapsulates various device constants, while the term I_{s0} depends on the transistor geometry (in particular, the aspect ratio of the transistor, W/L).

Immediately, this equation shows the dependence of leakage to W/L, and its exponential dependence to Vds, Vgs, Vth, and T.

1) W/L, transistor geometry: Leakage grows with the aspect ratio of a transistor and with its size. Butts and Sohi [13] use simplified models that encapsulate transistor geometry in the k_{design} parameter. They point out that very small transistors such as those found in SRAMs can leak much less than sized-for-performance logic gate transistors. Transistor sizing is primarily a circuit-level concern and it will not preoccupy us at the architecture level.

2) Vds, voltage differential between the drain and the source: This is probably the most important parameter concerning the architectural techniques developed for leakage. Two important leakage-control techniques that are based on reducing Vds are the transistor stacking technique [14] [15] and the drowsy technique [16]. Both these techniques rely on the $(1 - e^{-(\frac{Vds}{Vth})})$ factor of the sub-threshold leakage equation. This factor is approximately 1 with a large Vds (i.e., $Vds = Vdd$ and $Vdd >> vt$) but falls off rapidly as Vds is reduced.

3) Vgs, voltage differential between the gate and source: Regarding sub-threshold leakage for devices in their normal "off" state, this factor can be set to zero, so it is not a concern. Butts and Sohi use this assumption to arrive at their simplified leakage model [13]. However, Vgs plays a significant role in the gate-oxide leakage as discussed in next subsection.

4) Vth, threshold voltage: The threshold voltage – the voltage level that switches on the transistor – significantly affects the magnitude of the leakage current in the off state. The exponential dependence of sub-threshold leakage on $(Vth)^{-1}$ is evident in the last factor of equation2.3: the smaller the Vth, the higher is the leakage. Raising the threshold voltage reduces the sub-threshold leakage but compromises switching speed.

Many circuit-level techniques, e.g., MTCMOS, Reverse Body Bias (RBB) and larger-than-Vdd Forward Body Bias (FBB) [17] [18] [19] [20] [21], have been developed to provide a choice of threshold voltages. These techniques provide multiple threshold voltages at the process level (for example, MTCMOS offers high-Vth and low-Vth devices) or vary the threshold voltage dynamically by applying bias voltages on the semiconductor body (e.g., RBB and larger-than-Vdd FBB).

5) T, temperature: Last but not the least, sub-threshold leakage exponentially depends on temperature, T, via the thermal voltage term ν_t [22] [23] [24] [25]. This is actually a dangerous dependence since it can set off a phenomenon called thermal runaway [26]. If leakage power – or for that matter any other source of power consumption – causes an increase in temperature, the thermal voltage ν_t also increases linearly to temperature. This leads, in turn, to an exponential increase in leakage,

which further increases temperature. This vicious circle of temperature and leakage increase can be so severe as to seriously damage the semiconductor. The solution is to keep the temperature below some critical threshold so that thermal runaway cannot happen. Cooling techniques, combined with accurate thermal monitoring, are used for this purpose.

2.2.2 Gate Leakage

Gate leakage (also known as gate-oxide leakage) is a major concern because of its tremendous rate of increase. It grew 100-fold from the 130 nm technology (2001) to the 90 nm technology (2003) [9]. Major semiconductor companies are switching to high-k dielectrics in their process technologies to alleviate this problem [9].

Gate leakage occurs due to direct tunneling of electrons through the gate insulator – commonly silicon dioxide, SiO_2 – that separates the gate terminal from the transistor channel [27] [28]. The thickness, Tox, of the gate SiO_2 insulator must also be scaled along with other dimensions of the transistor to allow the gate's electric field to effectively control the conductance of the channel. The problem is that when the gate insulator becomes very thin, quantum mechanics allow electrons to tunnel across. When the insulating layer is thick, the probability of tunneling across it is virtually non-existent. As the insulating layer becomes thinner, tunneling becomes stronger. Gate-oxide thickness has scaled from 100nm to just 1.2 nm in 90nm and 65nm technologies. This corresponds to a thickness of just 4~5 atoms [29]! The result is an uncontrollable, exponential increase in gate leakage.

Gate leakage is somewhat dependent on temperature but strongly dependent on the insulator thickness and the gate-to-source (Vgs) or gate-to-drain (Vgd) biases seen by the device. Without the Vgs or Vgd biases, the necessary electric field to cause the electrons to tunnel across the gate is absent. Since the supply voltage determines the magnitude of Vgs and Vgd, scaling Vdd reduces gate leakage. There is also a weaker dependence of gate leakage on Vds – the voltage across the drain and source – that ties gate leakage to the state of a circuit [30].

The most promising remedy for gate leakage, and the one that is currently in use in the latest generation 45 nm technologies, is to insulate the gate using high-k dielectric materials instead of the more common SiO_2 oxide material. A thicker insulating layer of a high-k material can be as good as a thin layer of a low-k material. The increased thickness significantly reduces the tunneling effect but at the same time does not compromise the ability of the gate to control the channel. In other words, performance is not compromised.

Gate leakage has not been given the same attention as sub-threshold leakage at the architecture level or the circuit level. For the most part, it is considered as an additional leakage component and the hope is that process-level solutions will address the problem.

2.3 Circuit-level Leakage Control Techniques

This section briefly presents and compares several circuit-level leakage control techniques. Leakage has several different components, however the largest components are sub-threshold related [31]. As mentioned in last subsection, leakage control techniques focus on controlling one or more terms in equation2.3. The most prevalent techniques can be categorized as reducing Vgs, increasing Vth, lowering Vbs, and reducing Vds. Several different methods for controlling these terms are described below along with how they relate the equation.

1) Reverse Body Bias (RBB)

Since leakage currents are a function of the device thresholds, one method for controlling leakage is to control Vth through the use of substrate, or body, bias. In this case, the substrate or the appropriate well is biased so as to raise the transistor thresholds thus reducing leakage. Since raising Vth also affects performance, the bias can be applied adaptively such that during active mode the reverse bias is small while in standby mode the reverse bias is more negative. Thus, reverse body bias reduces leakage by increasing Vth due to decreasing the γVbs in equation2.5. Use of body bias requires a substrate bias-generator to generate the bias voltage. This generator will consume some dynamic power, partially offsetting any gain from reduced leakage. However, a more significant issue with the use of substrate biasing for leakage reduction is that it is generally less effective in advanced technologies [32]. The body effect factor γ decreases with advanced technologies [33], reducing the extent of the leakage control. Consequently, reductions of 4x at 90nm and only 2x at 65nm have been reported [34].

2) Dynamic Voltage Scaling (DVS)

One technique for reducing dynamic power, dynamic voltage scaling, can also be used for reducing leakage power. DVS works by reducing the power supply voltage Vdd when the work load does not require maximal performance. DVS can also be applied to inactive circuits for leakage reduction. In equation2.3, the reduction in Vdd is reflected in a smaller value for Vds which has an exponential effect on leakage. Power savings of 8x to 16x have been reported when scaling the voltage to the 300mV range, the lowest voltage at which state can be maintained [35]. However, DVS requires additional circuitry to monitor and predict the workload as well as a dynamic voltage regulator to dynamically adjust the supply voltage. Also, the timing analysis of DVS circuitry is complicated since proper operation must be validated over a number of additional voltage points. Nevertheless, DVS has been combined with RBB for even greater leakage reduction than either technique can achieve alone [36].

3) Multi-Vth Cell Swapping

The most prevalent technique used to date for leakage reduction is multi-Vth cell swapping, most commonly deployed with two different transistor thresholds (and hence known as dual-Vth cell optimization) [37] [38]. In this technique, low-Vth cells are used on critical paths while high-Vth cells are used on non-critical paths. The low-Vth cells are fast but leaky, while the high-Vth cells are

just the opposite. Thus, the multi-Vth technique can reduce leakage power without any performance penalty.

A significant advantage of multi-Vth cell swapping is that it is generally footprint neutral. That is, no floor planning or layout changes are required for implementation. High-Vth cells replace their low-Vth equivalents in exact positions in the layout, thus effectively changing only the implant mask. Leakage can typically be reduced by about 50% compared to a circuit implemented with all low-Vth cells although the reduction is heavily dependent upon the amount of available slack in the original circuit [39]. However, the remaining low-Vth cells still consume significant amounts of leakage power. Thus, this technique is usually insufficient for achieving large reductions in standby mode leakage power. For this reason, designers have turned to more aggressive leakage control design techniques such as MTCMOS power gating [40].

4) MTCMOS Power Gating

MTCMOS power gating is a design technique in which a power gating transistor is inserted in the stack between the logic transistors and either power or ground, thus creating a virtual supply rail or a virtual ground rail, respectively [41] [42] [43]. Such configurations are shown in Fig.2.2. The logic block contains all low-Vth transistors for fastest switching speeds while the switch transistors, header or footer, are built using high-Vth transistors to minimize the leakage. Power gating can be implemented without using multiple thresholds, but it will not reduce leakage as much as if implemented with multiple thresholds. MTCMOS refers to the mixture of the transistor thresholds in power gating circuits. The most common implementations of power gating use a footer switch alone to limit the switch area overhead. High-Vth NMOS footer switches are about half the size of equivalent-resistance high-Vth PMOS header switches due to differences in majority carrier mobilities.

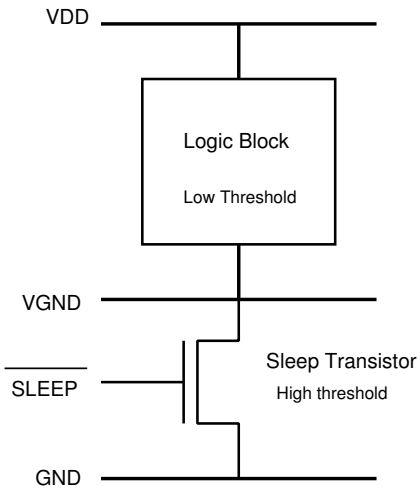


Figure 2.2: Power Gating Structure (footer switch)

Power gating reduces leakage by reducing the gate-to-source voltage which in turn drives the

logic transistors deeper into the cutoff region. This occurs because of the stack effect. The source terminal of the bottom-most transistor in the logic stack is no longer at ground, but rather at a voltage somewhat above ground due to the presence of the power gating transistor. Leakage is reduced due to the reduction of the Vgs.

Power gating itself has several variants, such as Super Cut-off CMOS [44] and Zigzag Super Cut-off CMOS [45]. In Super Cut-off CMOS, instead of using high-Vth NMOS or PMOS switch transistors, low-Vth switch transistors are used. In standby mode, the switches are driven deeper into cut-off by applying a gate voltage below Vss for NMOS switches and above Vdd for PMOS switches, thus decreasing Vgs beyond what can be achieved with conventional gate voltages. In Zigzag Super Cut-off CMOS, both header and footer switches are used in an alternating fashion along logic paths in combination with super cutoff CMOS to reduce the amount of time required for the virtual rails to settle after turning on the switch transistors.

Power gating can be combined with other leakage reduction techniques, such as those described above, to achieve even greater leakage reduction. When implemented alone, power gating can achieve 100~1000x reduction in leakage [46]. When implemented in combination with other techniques, such as reverse body bias on the switch, the reduction can be even larger.

2.4 Architecture-level Leakage Control Techniques

Undeniably, the most fruitful ground for developing leakage-reduction techniques at the architectural level has been the cache hierarchy [47] [48] [49] [50] [51] [52] [53] [54] [55] [56] [57] [58] [59]. The large number of transistors in the on-chip memory largely justifies the effort (or obsession) even though these transistors are not the most "leaky" – that distinction goes to the high-speed logic transistors [13]. In addition, the regularity of design and the access properties of the memory system have made it an excellent target for developing high-level policies to fight leakage. In this section, we present three representative architecture-level leakage control techniques on the cache memory.

2.4.1 Cache Decay

The technique, called cache decay, turns a cache line off if a pre-set number of cycles have elapsed since its last access [47]. However, turning off a cache line that is in active use incurs extra dynamic power re-fetching the line from lower cache levels. Therefore, a central goal of cache decay is to accurately predict when a cache line is no longer useful, or – as it is more commonly known – when a cache line is dead.

2.4.1.1 Generational Behavior

Cache decay is based on the fundamental generational behavior of cache lines, which was first discovered by Wood, Hill, and Kessler in their effort to support faster trace driven cache simulation [60].

A generation begins when a cache line is brought into the cache after a miss. Immediately after the miss, a flurry of access activity ensues. While the cache line is being accessed, it is in its live time. At some point, the accesses cease and the cache line sits idle waiting to be evicted to make room for a new line. The time spent in this state is called the dead time. The fundamental characteristic of the generational behavior of cache lines, the reason why most of the cache is "empty" of useful data, and the reason why unknown references in trace sampling have a very high miss rate, is that (on average) the dead time of cache lines by far exceeds their live time. Simulations with SPEC2000 benchmarks show that total dead time (for all cache lines) accounts for about 80% of the total time, on average. This fundamental characteristic is exactly what cache decay tries to exploit.

Cache decay tries to guess whether a cache line is live at any particular point in time. It does this by measuring elapsed time since the last access to the cache line. Since the cache line is accessed only during its live time – which is typically short – it follows that the inter-access time between two consecutive accesses should be particularly short. It is easy then to guess when a cache line is not in its live time: if sufficient time has passed without an intervening access then most likely the cache line has entered its dead time and awaits eviction.

The main task of cache decay is to gauge the idle time of a cache line in relation to its inter-access times. When the idle time of a cache line exceeds a limit called the decay interval (which is set to be

beyond the cluster of the small inter-access times), the cache line is predicted to be in its dead time and is shut off using a gated-Vdd sleep transistor.

2.4.1.2 Implementation

While there are a few possible implementations for cache decay (including some analog varieties), one of the better known methods uses a scheme of hierarchical counters [47]. The idea is to use counters in the cache lines to measure their idle time. A counter works like a stopwatch: it starts ticking after an access; if the cache line is accessed, it is reset; if, however, it ticks uninterrupted until it reaches the decay interval then the cache line is pronounced "dead". A counter maximum in the thousands would incur too much overhead, however, to include with each and every cache line.

The solution is to use much smaller, coarser-grain counters in the cache lines. These small counters advance every few hundreds (or even thousands) of cycles rather than every single one. The beat is given by a single global cycle counter which counts these larger intervals. So, for example, if the global counter counts 1024 cycles and the local cache line counters are 2 bits, then they count decay intervals up to 4×1024 cycles. This scheme minimizes overhead, since the global counter can be easily piggybacked on cycle counters commonly found inside processor cores and the local cache line counters can be implemented asynchronously to minimize switching overhead. The high-level view of the cache decay implementation and the hierarchical counters are presented in Fig.2.3

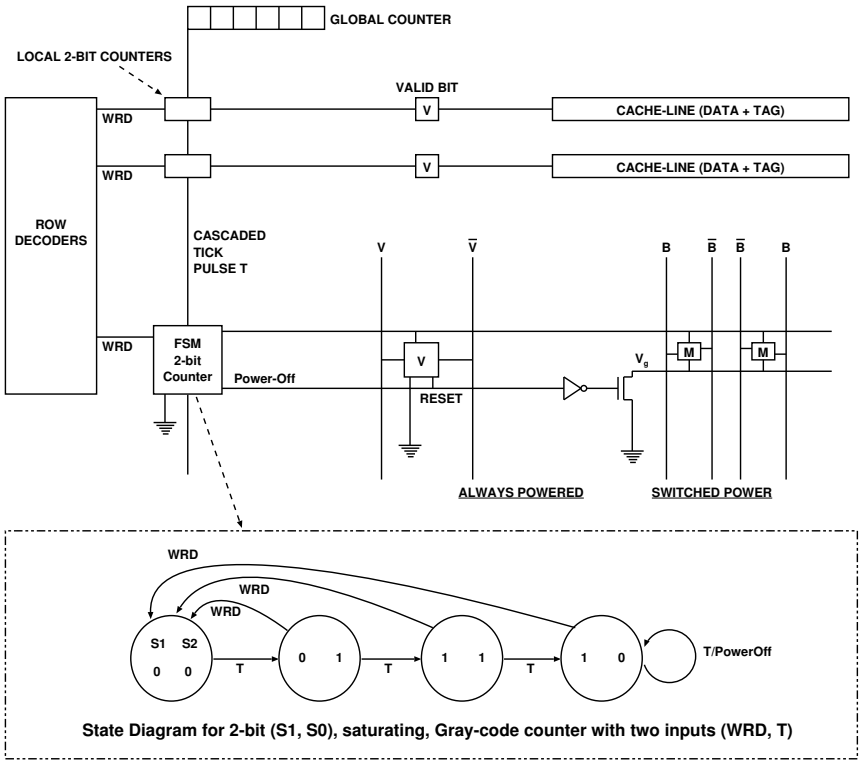


Figure 2.3: Cache Decay Implementation

2.4.1.3 Results

Cache decay has been extensively tested using the SPEC2000 in many cache configurations (e.g., instruction, data, L1, L2, direct mapped, set associative, and for many cache sizes). Overall, decay is very successful in switching off a significant part of the cache, on the order of 70% for Level-1 caches, impacting a minimal performance penalty of a few percent (less than 4%).

However, decay can also result in energy and performance penalties. Because of the destructive nature of the power-gating mechanism, mistakenly switching off a cache line results in an additional L2 access. These are called decay-induced misses and cost not only in energy (reducing the energy benefit of decay) but also in performance.

Further results can be found in Hanson's work with Hrishikesh, Agarwal, Keckler, and Burger. Hanson's work is one of the most detailed and extensive studies on cache decay and provides comparisons with two other leakage-saving techniques for caches. A detailed technical report by Hanson et al. greatly expands on the initial results reported for decay [61].

One disadvantage of the power-gating mechanism is that it destroys state. The first approaches to control leakage based on this mechanism [47] [51] are known as non-state-preserving. In the case of cache decay and related approaches, the reasoning is that most of the lost state is useless anyway. And that would be fine if it were not for the problem of mistakes – decay-induced misses – which actually harm performance. The drowsy effect was proposed to address this problem, introducing a new class of state-preserving leakage-reduction techniques.

2.4.2 Drowsy Cache

In response to the problem of losing state caused by power gating, Flautner, Kim, Martin, Blaauw, and Mudge proposed another approach to curb leakage in memory cells [48]. The drowsy mode is a low supply voltage mode for the memory cells, i.e., Dynamic Voltage Scaling (DVS) for leakage.

Similar to the DVS approaches discussed in subsection 2.3, this type of DVS also has to do with idleness; but without frequency scaling. Memory cells which are idle, i.e., are not actively accessed, can be voltage-scaled into a drowsy mode. In this mode, transistors leak much less than with a full Vdd. Fig.2.4 shows the design of a drowsy cache. A "drowsy" bit controls the two levels of supply voltage (Vdd or VddLow) to the memory cells of a cache line. Memory cells are in drowsy mode when fed from VddLow.

The leakage reduction of the drowsy mode is not as profound as that of the gated-Vdd approach which completely cuts off the path to Vdd (or, equivalently, to ground). However, allowing for some nonzero supply voltage preserves the state of the memory cell. This happens as long as the supply voltage is strong enough to replenish the charge in the cell's internal nodes. However, a memory cell in drowsy mode cannot be accessed with the full-Vdd circuitry of the cache. It first has to be voltage-scaled back to full Vdd. Because this is not instantaneous, there is a penalty, albeit small, in accessing drowsy cells.

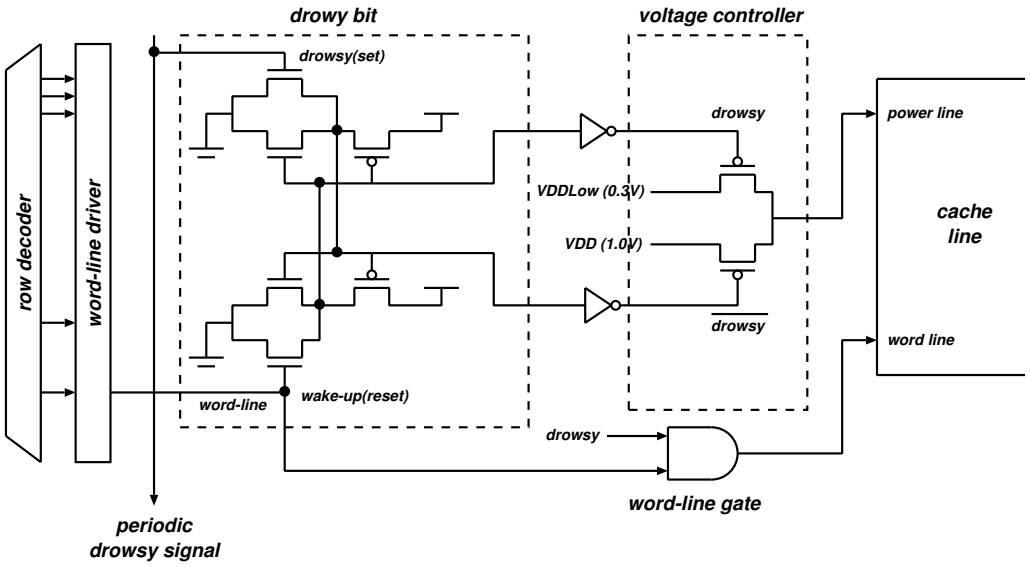


Figure 2.4: Drowsy Cache Implementation

2.4.2.1 Implementation and Results

Because state is preserved in drowsy mode, there is no danger in experiencing long miss latencies when accessing drowsy cache lines. The penalty to voltage-scale a drowsy cache line back to full Vdd is relatively small – a few (single-digit) cycles. Whereas it would matter significantly which cache lines are put into low-leakage mode in a nonstate-reserving technique, with the drowsy mode it does not matter; mistakes cost very little. This makes sophisticated techniques that determine the idleness of cache lines unnecessary, especially if one factors in their dynamic power cost. Flautner et al. thus propose a very simple policy – fittingly called Simple – for the drowsy mode: the whole cache is periodically put into drowsy mode – all of the cache lines regardless of usefulness or idleness. The small percentage of active cache lines are going to exit the drowsy mode, on demand, incurring a small latency penalty. Since this latency is experienced on hits, programs which are sensitive to hit latency are going to be hurt the most. A variable hit latency can also complicate instruction scheduling in an out-of-order core, further degrading the performance [62].

The simple policy is quite effective: it can put into drowsy mode 80% $\sim$ 90% of a 32KB L1 data cache while incurring a slight performance penalty of 1%. These numbers are for a four-instruction wide out-of-order core and assume a very aggressive one-cycle penalty for accessing drowsy cache lines. The Simple policy does not perform as well with instruction caches which need to be handled differently.

2.4.2.2 Improvements on the drowsy policy

Petit, Sahuquillo, Such, and Kaeli [63] improved on the Simple policy by applying few smart heuristics. Their approach is to maintain the low complexity of the initial idea by adding very little hard-

ware. The goal is to improve on the Simple policy which blindly puts all cache lines in drowsy mode. In the Simple policy no effort is spent to distinguish between active (important) and idle (useless) cache lines. On the other hand, precisely determining the individual status of each and every cache line, la cache decay, veers off the desired course of simplicity. Instead, Petit et al. propose simpler heuristics to filter the lines that are put in the drowsy mode. Considering the cache lines in an associative set, it is obvious that if there is a live cache line among them it must be the most recently used (MRU) line. Thus, the first policy is to exclude the MRU line from going into drowsy mode. In fact, the policy allows only the MRU line to remain awake in the set. The policy is called MRU ON, or simply MRO. Upon a hit on a drowsy line, the line is woken up and becomes the MRU line. The previous MRU line is put in the drowsy mode. According to Petit et al., 92% of the cache hits are hits on the MRU line so a change in the MRU status, which incurs dynamic switching, is rare.

The second policy simply keeps awake the two most recently used lines per set in as much as both are good candidates for being active (alive) rather than idle (dead). The policy is called Two-MRO (TMRO). This choice is justified because most of the remaining 8% of the hits that do not hit the MRU line are accesses to the second most recently used line. Since more lines per set are kept awake energy savings are less than MRO but accesses to drowsy lines are minimized.

2.4.3 Hotspot Detection on Instruction Caches

The compiler can also provide application-sensitive leakage control. For example, a program 's source code could include explicit loop-level cache line turn-off instructions. Researchers at the Pennsylvania State University have developed a scheme to manage instruction cache leakage that is sensitive to changes in temporal and spatial locality during program execution [50]. The scheme builds on drowsy cache techniques. It associates each cache line with a mode bit that controls whether the line is awake and accessible or in low-leakage drowsy mode. Periodically, a global sleep signal resets all the mode bits to put all the cache lines in drowsy mode. This approach is based on the notion that working sets change periodically. In reality, the change is gradual, so asserting the sleep signal will unnecessarily put some lines that are part of the working set into drowsy mode.

2.4.3.1 Identifying hotspots

One improvement on this basic idea prevents inadvertent mode transitions by augmenting each cache line with a local voltage-control- mask bit. When set, the VCM masks the influence of the global sleep signal and prevents mode transition. The VCM bits are set based on information from an enhanced branch target buffer. The BTB monitors how often an application accesses the different basic blocks and uses this information to identify whether they belong to a program hotspot. Once the BTB determines that a program is within a hotspot, the processor sets a global mask bit, then sets the VCM bits of all accessed cache lines to indicate the program hotspot. The processor updates the BTB access-frequency counters and the VCM bits periodically to reflect the change in program

phase.

2.4.3.2 Predicting transitions

A second improvement focuses on predicatively transitioning the cache lines that an application program will access next from sleep to normal mode. The predictive strategy avoids the performance and associated leakage penalty incurred from accessing a cache line in sleep mode. Since sequentiality is the norm in code execution, the program counter predicatively transitions the next cache line to the normal mode when it accesses the current line. This technique is referred to as just-in-time activation.

2.4.3.3 Tracking access moves

Finally, asserting the global sleep signal when accesses move from one cache sub-bank to another enables the processor to identify opportunities for sleep mode transition when spatial locality changes. This differs from asserting the global sleep signal periodically to capture temporal locality changes.

2.4.3.4 Performance improvements

When averaged across 14 SPEC2000 benchmarks, these three improvements provide an average leakage energy savings of 63% in the instruction cache compared to using no leakage management, 49% compared to the bank-based turnoff scheme, and 29% percent over the compiler-based turnoff scheme.

2.5 Conclusion

In this chapter, we briefly introduced the underlying mechanics of leakage power and the circuit-level leakage reduction techniques. In addition, three representative leakage-control mechanisms on caches, which combine the circuit-level techniques and efforts on architecture-level, have also been illustrated. Although theses mechanisms can achieve a significant leakage power reduction on the on-chip caches, to ensure the overall leakage-reduction effect of the whole processor, leakage-reduction mechanisms on other on-chip components are also necessary. For such a purpose, in Chapter 3, a comprehensive leakage analysis of embedded processors is presented; and based on the analysis result, we pick up three components (functional units, iTLB, and dTLB) as the leakage-reduction targets. Then, in Chapter 4, Chapter 5, and Chapter 6, leakage-reduction mechanisms on theses components will be illustrated with combination of the circuit-level techniques presented in Section 2.3.

Chapter 3

Embedded Processor Leakage Analysis

3.1 Introduction

Low-leakage processor design requires an in-depth examination of each on-chip component, and in the chapter, a detailed leakage analysis of embedded processors is presented. However, there are more than 100 vendors and two dozen instructions set architecture in the 32-bit embedded-processor market [64], and selecting a architecture, which can represent the essential structure of all embedded processors, may be difficult. The leakage analysis in this chapter is based on a MIPS R3000 processor [65] [8]. Before going to the details of leakage analysis, we first discuss the trends of modern embedded-processor design and the strategy of the architecture-selecting of this chapter.

While it is commonly accepted that embedded processor are playing an important part in our daily life, the term of embedded processor is nebulous and to some it refers to just about everything except processors found in desktop computers, workstations, servers, and mainframes. In the past, embedded processor is usually referred to as processors that are designed for a specific function, implying that embedded processors execute only one program, repeatedly [66]. Thus, one type of embedded processors are usually highly optimized for one application, and they usually can achieve the best efficiency in terms of performance, power and cost for the target application. At this scenario, the leakage optimization must base on each specific type of embedded processors, and finding a representative architecture may be a mission impossible.

However, the ever-changing market requests alter the morphology of embedded-community, and we are witnessing a path to increased standardization and unification of embedded processor architecture [64]. New applications and market segments create the economic demand on the new and specialized devices; on the other hand, cost and time-to-market considerations make it mandatory to reuse these devices in other applications, avoiding the steep learning curves that accompany product development. For these reasons, recently, most companies design embedded processors not only for a very specific market but for the more general one. Thus, in some ways, modern embedded processors begin to adopt more unified structures, with integration of extension components or even hardware accelerators which are optimized for a specific kind of applications.

Such a new design trend gives us an opportunity to study the leakage consumption of each on-chip components in a more general way – we only concern about general components which may be employed in many different architectures, but omitting those that are optimized for individual application.

In this chapter, the MIPS R3000 processor is selected for the purpose of leakage analysis not only because it employs a simple structure but also because it includes almost all essential on-chip components of modern embedded processor – pipeline, caches, memory management unit (MMU), and so on. In the following chapters, such a processor is also used as the experimental infrastructure to measure the efficiency of proposed leakage-reduction mechanisms. Note that, although the leakage-reduction effects in this thesis are based on the MIPS R3000 architecture, proposed mechanisms can be generically applied to other architectures with minor modifications according to various specifications. To better understand the mechanisms proposed in following chapters, before going to the leakage-analysis results, the specification and structure of MIPS R3000 are introduced first in this chapter.

3.2 MIPS R3000 Features and Specifications

3.2.1 Reduced Instruction Set Computing

MIPS is the most elegant among the effective Reduced Instruction Set Computing (RISC) architecture [67]. When the first processors and Instruction Set Architectures (ISAs) were created, programming was very difficult, and complex instructions that looked more like high level languages were added to each ISA family. To allow old software to run on newer computers the instruction sets were expanded and became more and more complex. Moreover, memory prices were very high and to fit a program into a very small space, it was necessary to create instructions with varying lengths. Also memory was a lot faster than the processor and so keeping all variables in memory was quite logical. In a completely different direction to such a Complex Instruction Set Computing (CISC) architecture, which increasingly complicates the ISA, RISC adopted a totally different approach. By keeping all instructions the same size decoding becomes simple. Having a large register bank decreases memory accesses which are increasingly slow compared to the processor speed on modern computers. Implementing only simple and common operations the processor speed can be increased, and chip area can be decreased. Having a much simpler base architecture allows the processor to implement other speed-increasing methods much more simply for pipelining.

3.2.2 Load Store Architecture

MIPS ISAs only allow registers or small immediates to be operands of operations. CISC, on the other hand, often has instructions which use memory stored data as operands, which makes the execution harder. For example, the 8086 instruction “CMP AX, ES:[SI+02]” firstly requires 2 to be added to SI, and the result to be added to ES shifted by 4, which creating an effective address to load a 16 bit word (possibly non-word aligned so multiple loads are required) from memory; then compare it to AX and write the flags created by the comparison to the flags register for use by the next conditional jump instruction. This scheme is difficult to implement, and the instruction goes several times through the Arithmetic/Logic Unit (ALU). MIPS instructions only go through the ALU once and never after a memory access. If an operand from memory is required, it is loaded into the register bank first and only then used in subsequent operations. This allows the creation of a very simple architecture which is easy for implementation. A typical MIPS processor uses a five stage execution pipeline as shown in Fig.3.1, which includes the Instruction Fetch stage (IF), the Instruction Decode stage (ID), the Execution stage (EX), the Memory Access stage (MEM), and the Write Back stage (WB). In a five stage pipeline, there are two memory components: one for instruction fetch and one for data accesses.

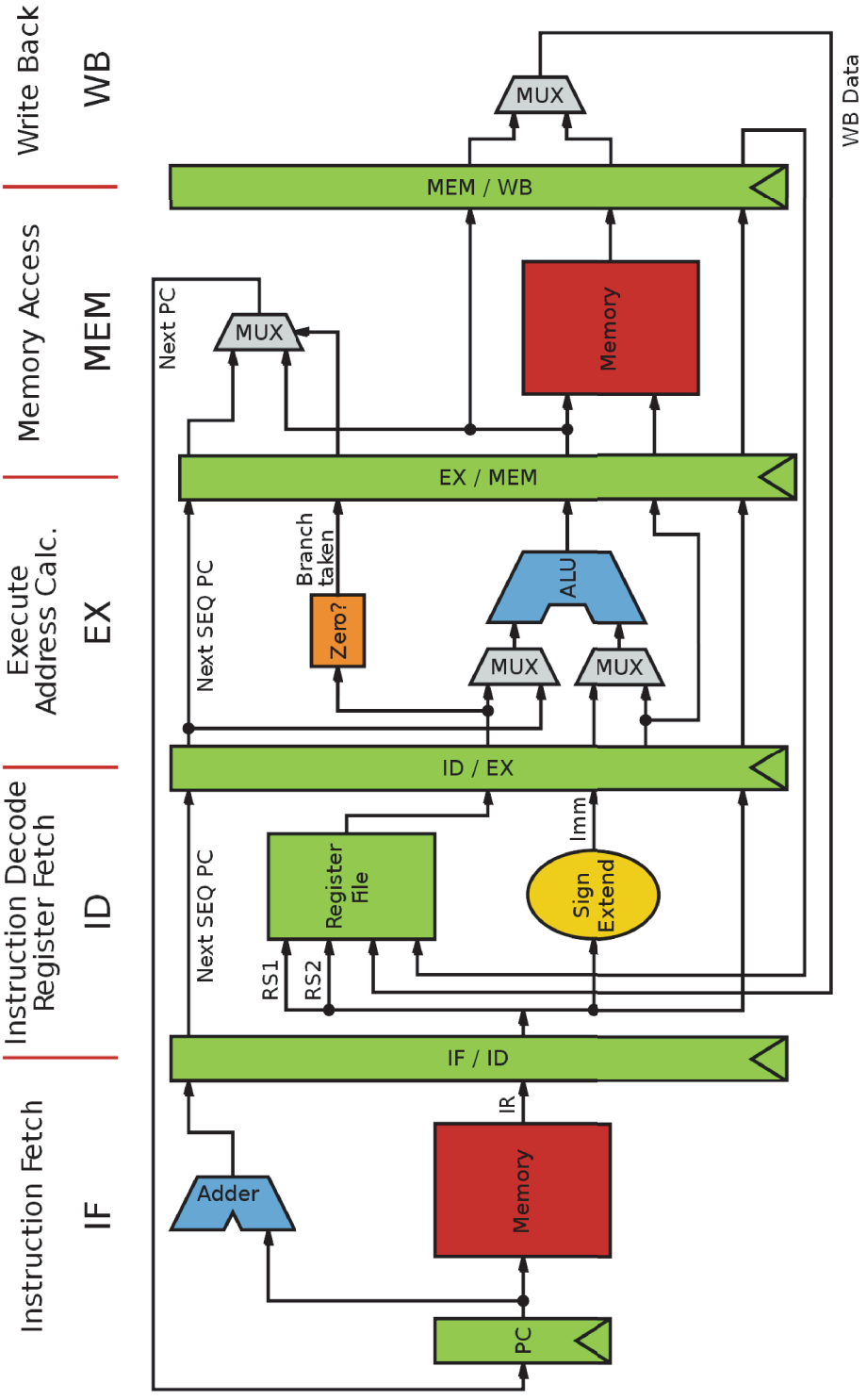


Figure 3.1: MIPS R3000 Pipeline Structure

3.2.3 Pipelining

Pipelining is a method of getting more than one instruction to execute simultaneously. By dividing the path that the instruction has to go through in the processor into segments and placing latches at the beginning of each segment, instructions will take several clocks to execute instead of one. Since MIPS instructions only visit each segment once, they only occupy one segment – allowing other instructions to come straight after them and occupy other segments. There are problems that arise with pipelining. If an ALU instruction writes to a register that is required in the next instruction, data in the register bank is not yet updated when the second instruction requests it as the data is now at the end of the EX stage. The easiest way of getting the data back to the next instruction is to forward the result from the EX stage, and replace the register bank value with it. The same can be done for data that is at the end of the MEM stage. This still does not solve the problem of using a result from a memory operation on the next cycle. Such a problem can be solved by the processor inserting a NOP instruction, if it detects a dependency or the compiler simply never using a result from a memory operation on the next cycle.

3.2.4 Instructions

All MIPS R3000 instructions are 32 bit and come in three formats – R-type, I-type and J-type. MIPS R3000 instructions are three address operations, taking two sources and one destination. The R-type instructions allow a range of register to register operations. The I-type instructions allow a 16 bit immediate to replace one of the operands. The I-type instruction format is also used for memory accesses and for conditional branches. The J-type format has a 26 bit immediate field, and the only instruction to use this format is a jump which places the value in the bottom 26 bits of the program counter.

3.2.5 Registers

A MIPS R3000 processor has 32 addressable registers. Register zero (R0) is special as it is always equal to zero, and writes to it are ignored. R31 is a normal register but when executing any branch or jump with store return address, the next PC is stored in R31. In addition to the addressable registers there are three more implemented registers. The Program Counter (PC) is not a part of the main register bank. It is accessible directly through Jump to Register (JR) for writing and Branch And Link (BAL) for reading. The other two registers are LO and HI. These registers are used for the results of the multiplier and divider. Although these can also be also accessed directly by Move To and From LO and HI instructions. All these registers are 32 bits wide although the bottom two bits of the PC should always be zero.

3.2.6 Conditions

There are no condition flags, but instead all branches are conditional on the values of the registers in the main register bank. Each conditional branch instruction specifies two registers (RS and RT) to be fetched and tested. A branch is conditional on the results of two tests. The first is compare the two registers together to test whether they are equal (RS=RT). The other test is simply to look at the sign value (bit 31) of the first register (RS<0). By choosing the second register to be R0 (RT=0), its becomes possible to test RS for less than greater or equal to zero or any combination of the three. For an unconditional branch the Branch if Greater or Equal to Zero instruction (BGEZ) is used with R0 as an operand. This condition will allays be true.

3.2.7 Memory

Memory access instructions are included in the I-type format. The source register (RS) is added to the immediate to create an effective address, which is used to reference the memory. The second register (RT) is either used as the destination in a memory load or as a source in a memory store. The memory is byte addressed but is 32 bit wide so all word loads and stores have to be word aligned. Half word accesses have to be aligned to half word boundaries. To help with unaligned loads and stores there are two more memory access instructions. Load Word Left (LWL) and Load Word Right (LWR) in combination allow word loads from unaligned addresses.

3.2.8 Pipeline Interlocking

MIPS stands for “Microprocessor without Interlocking Pipeline Stages”. In the MIPS processor this means that some instructions have an implicit delay before their effect takes place (This is not strictly true as the multiplier/divider has interlocking). The general philosophy is to construct the hardware as simply as possible and, if a result is not ready for use in the next instruction then not to stop the whole processor but use the software to insert instructions into the space. The two main delays in the MIPS processor are branch shadows and memory load delays. There are others but they happen very rarely.

3.2.8.1 Delayed Branch

When a branch is executed, the PC is only updated at the end of the next instruction. This is because the MIPS designers were using a pipeline that loaded the next instruction from memory while decoding the current. By the time the current instruction is decoded, and the processor detects it as a branch the next instruction is already loaded. The PC is updated by the time the next instruction after that is loaded. The delayed slot is filled with a useful instruction that the branch is not dependent on. If this instruction can not be found, then a NOP (Do nothing) instruction is placed to fill the entry.

3.2.8.2 Load Delay

Before a load can complete, the address must be calculated and then the load from memory can begin. As this uses two cycles, the result is not ready for the next instruction to use as at the time it wants the value the instruction has only calculated the address it is about to access. Again, there is an empty entry into which a useful instruction can be inserted if possible.

3.3 Structure of MIPS R3000 Processors

For the purpose of leakage evaluation, we have implemented a MIPS R3000 processor with Fujistu's 65nm CMOS process. The main structure of the implemented processor is presented as follows.

3.3.1 Pipeline

As shown in Fig.3.1, the execution of a single instruction consists of five primary pipe stages:

- IF – Instruction Fetch. Access the TLB and calculate the instruction address required to read an instruction form the instruction cache. Note that the instruction is not actually read into the processor until the begin of ID stage.
- ID – Instruction Decode. Read any required operands from processor registers while decoding the instruction.
- EX – Execution. Perform the required operation on instruction operands.
- MEM – Memory Access. Access memory (data cache) if required.
- WB – Write Back. Write back execution results or value loaded from data cache to register file.

An important implementation consideration of the pipeline structure is the non-interlocking properties as mentioned in last section. First, examining the fact that consecutive ALU instructions have no delay means that some form of forwarding is probably taking place. The memory loads have to take the result from the EX-stage and then pass the calculated address to the memory. The result is ready for use on the next cycle so again a forwarding scheme must be used. The branch shadow means that instruction prefetch happens irrespective of the instructions executed. The fact that branch shadow is only one cycle deep means that the PC must be updated within of one cycle of the instruction entering the pipeline. By using these rules it is possible to construct a simple pipeline that fits the non-interlocking requirements.

There is another problem with this pipeline. The branch unit requires results of the tests on values of registers. Although it is possible to read the registers and test if they match the branch conditions within the decode cycle, they will be the values from the register bank rather than the data from the forwarding paths. If the code affected the register in one of the three instructions previous to the branch conditional on this register then the register bank would not be updated. To deal with this the ID-stage is cut down to half a cycle. In this case, it is possible for the branch to complete within one cycle of the instruction fetch but still get forwarded values only available at the beginning of the EX-stage. By also decreasing the write back stage down to half a cycle it is now possible to use a register bank that does not need to have the ability of loading and storing simultaneously.

3.3.2 Cache

To reduce the average time to access memory, the implemented processor integrates on-chip L1 instruction cache and data cache.

The cache is a smaller, faster memory which stores copies of the data from the most frequently used main memory locations. As long as most memory accesses are cached memory locations, the average latency of memory accesses will be closer to the cache latency than to the latency of main memory. When the processor needs to read or write a location in main memory, it first checks whether that memory location is in the cache. This is accomplished by comparing the address of the memory location to all tags in the cache that might contain that address. If the processor finds that the memory location is in the cache, we say that a cache hit has occurred; otherwise, we speak of a cache miss.

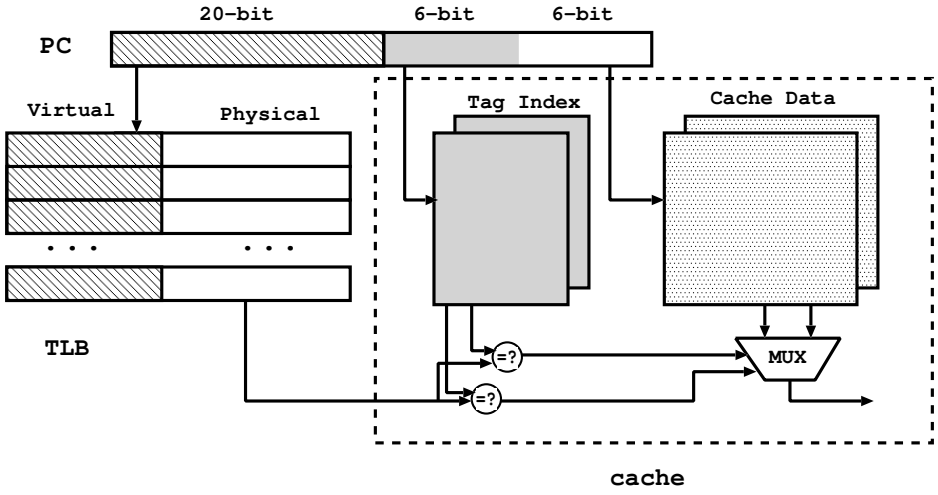


Figure 3.2: Cache Structure

Fig.3.2 shows the implemented cache structure. The size of both the instruction cache and the data cache is 8K bytes. The length of each cache line is 64 bytes. To reduce the ratio of the conflict cache misses [68], 2-way set-associative structure has been adopted on both cache. The data cache employs the write back policy when store instructions are executed, implying that the information is written only to the block in the cache, and the modified cache block is written to main memory only when it is replaced. Furthermore, both the instruction cache and data cache are the virtual-index physical-tag one, meaning that the TLB and cache Tag can be accessed simultaneously which reduces the length of the circuit path in the ID-stage.

3.3.3 Functional Units

Computational instructions, which is calculated in the EX-stage, perform arithmetic, logical and shifter operations on value in registers. As shown in Fig.3.3, in our MIPS R3000 processor, four

functional units have been implemented.

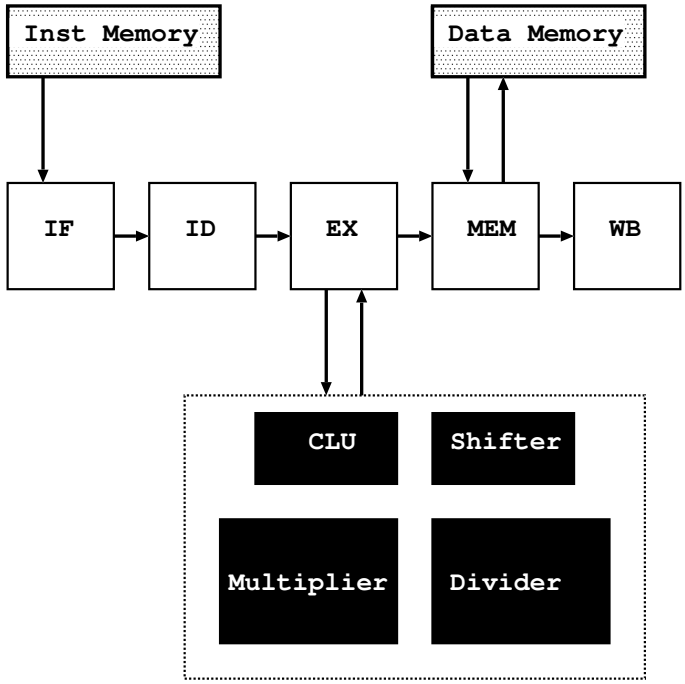


Figure 3.3: MIPS R3000 Functional Units

The four functional units include:

- CLU (Common arithmetic and Logic Unit)
  A general computational unit for addition, subtraction and all logical operations instead of the shift operation.
- Shifter
  A barrel shifter which can shift a data word by a specified number of bits in one clock cycle.
- Multiplier
  A 32-bit multiplier which takes 4 clock cycles for each multiplication operation.
- Divider
  A 32-bit Divider which takes 10 clock cycles for each division operation.

3.3.4 TLB

The Translation Lookaside Buffer (TLB) is a cache structure that memory management hardware uses to improve virtual address translation speed. Since the TLB is one of the leakage-reduction target presented in this thesis, to better understand the leakage-reduction mechanisms (Chapter 5 and Chapter 6), the principle and implementation of the TLB are illustrated here.

3.3.4.1 Virtual Memory

Any instant in time computers are running multiple processes, each with its own address space. It would be too expensive to dedicate a full address space worth of memory for each process, especially since many processes use only a small part of their address space. Hence, there must be a means of sharing a smaller amount of physical memory among many processes. It is important for virtual memory schemes like the follows: (a) Virtual memory, divides physical memory into blocks and allocates them to different processes. Inherent in such an approach can provide a protection scheme that restricts a process to the blocks belonging only to that process. Most forms of virtual memory also reduce the time to start a program, since not all code and data need be in physical memory before a program can begin. (b) Without virtual memory the programmer has to ensure that the program never tried to access more physical main memory than was in the computer, the virtual memory scheme can relieve the programmers of this burden, by automatically manages the two levels of the memory hierarchy represented by main memory and secondary storage. (c) Virtual memory also simplifies loading the program for execution which called relocation. This scheme allows the same program to run in any location in physical memory.

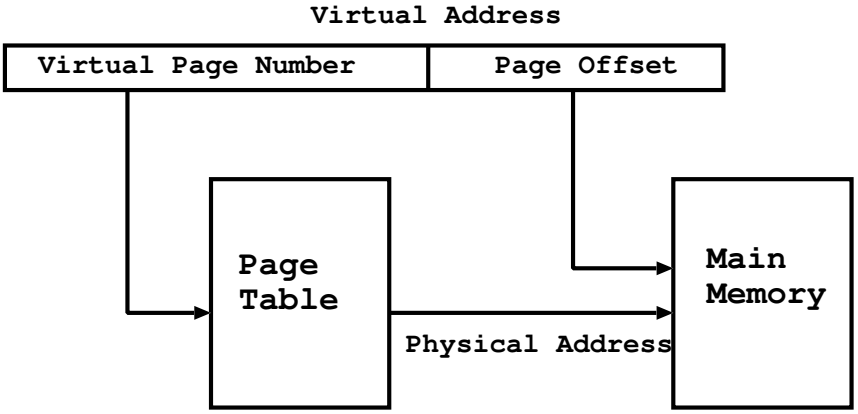


Figure 3.4: Virtual Address to Physical Address via Page Table

In order to use virtual memory, the paging scheme has been introduced. Usually the page size is 4K-byte for the embedded small system, the virtual translate to main memory is relied on a data structure which is called page table. Indexed by the virtual page number, and the offset is simply concatenated to this physical page address as shown in Fig.3.4. A 32-bit virtual address, 4KB pages will need 4MB page table, to reduce the size of this data structure and in order to reducing address translation time, computers use a cache desiccated to these address translations, called TLB, described in more detail below.

3.3.4.2 The Benefit of TLB

The essence of memory manager is to provide each program with its own memory space. So a process can crash or misbehave without bringing down the whole system. Any system in which you may want to run different programs at different times will find it easier if it can map the program's idea of addresses onto whatever physical address space is readily available. TLB is a kind of cache take the burden of address translation task. If you're using a full-scale OS like Linux, the TLB will be used behind and you'll rarely notice. The TLB mechanism makes it possible for translating addresses (at page granularity) from any mapped address to any physical address and therefore to relocate regions of program space to any location in your machine's address map. The TLB also allows you to define some address as temporarily or permanently unavailable, so that accesses to those locations will cause an exception that can be used to run some operating system service routine. By using user-privilege programs you can give some software access only to those addresses you want it to have, and by using address space IDs in the translation entries, you can efficiently manage multiple mutually inaccessible user programs. You can write-protect some parts of memory. The main idea is that the TLB, with all the ingenuity of a specification that fits so well into a big OS, is a useful, straightforward general resource for programmers. Even for embedded usage TLB is also required.

3.3.4.3 TLB Entry Structure in MIPS R3000

In the implemented MIPS R3000 processor, a 16-Entry mixed instruction/data TLB has been integrated. Each entry in the TLB has the virtual address of a page (the VPN for virtual page number), and a physical page address(PFN for page frame number). When a program presents a virtual address, it is compared with each VPN in the TLB, and if it matches an entry the corresponding PFN is delivered. A set of flag bit is stored and returned with each PFN and allows the OS to designate a page as read-only or to specify how data from that page might be cached. Fig.3.5 shows one TLB entry with the bit number. The total bits number is 64, the VPN and PFN owns 20-bit respectively.

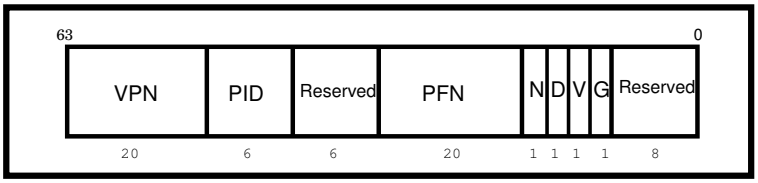


Figure 3.5: TLB Entry Structure(Unit:Bit)

In MIPS R3000, all TLB entries being written or read are staged through the registers EntryHI and EntryLO which are maintained by the coprocessor0. These two registers provide the data pathway through which the TLB is read, written, or probed. When address translation exceptions occur, these registers are loaded with relevant information about the address that caused the exception. The format of the *EntryHI* and *EntryLO* register pair is the same as the format of TLB entry. The meaning

of the items in each entry are as follows:

- *VPN*: The virtual page number.
- *PID*: The process identifier which is normally left holding the operating system's idea of the current address space. It is to mark the translation as belonging to a particular address space, so this entry will only be matched if the thread presenting the address has PID set equal to this value.
- *Reserved*: Currently ignores writes, returns zero when read.
- *PFN*: The physical page number.
- *N*: It denotes noncacheable that if the bit set to 1, means the processor directly accesses main memory instead of first accessing the cache. The field has been used in CPUs aimed at embedded applications, when it selects how the cache works: for example, marking some particular page to be managed "write-through" that is, all writes made there are sent straight to main memory as well as to any cached copy.
- *D(dirty)*: if this bit is set, the page is marked as "dirty" and therefore writable. This bit is actually a "write-protect" bit that software can use to prevent alteration of data. If an entry is accessed for a write operation when the D bit is cleared, the processor causes a TLB Mod trap. The TLB entry is not modified on such a trap.
- *V(valid)*: If this is 0, the entry is unusable and any use of an address matching this entry will cause an exception.
- *G(global)*: If set to 1, disables the PID match, making the translation entry apply to all address spaces that is this part of address map is shared between all spaces.
- *Reserved*: Currently ignores writes, returns zero when read.

3.3.4.4 Virtual Address Translation Flow in MIPS R3000

The virtual address is translated by TLB and the physical address is outputted. Fig.3.6 shows the form of processor virtual address. The first 3 most-significant bits present the address mode which will be explained below.

When the processor is operating in User mode, a single, uniform virtual address space (kuseg) of 2 Gbytes is available for users. All valid User-mode virtual addresses have the most-significant bit cleared to 0. An attempt to reference an address with the MSB (most significant bit) set while in the User mode causes an Address Error exception. The 2 Gbyte User segment starts at address zero. The TLB maps all references to kuseg identically from Kernel and User modes, and controls access cacheability (The N bit in the TLB entry determines whether the reference will be cached.).

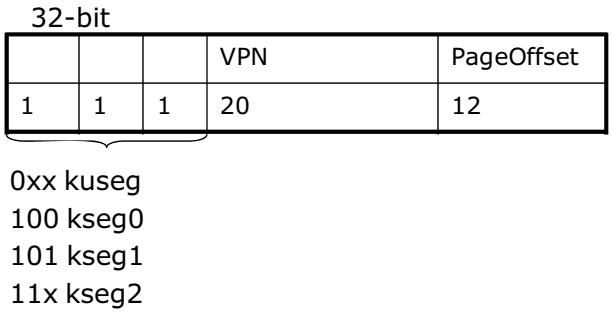


Figure 3.6: Input Virtual Address Structure

The kuseg is typically used to hold user code and data, and the current user process typically resides in kuseg.

When the processor is operating in Kernel mode, three distinct virtual address spaces are simultaneously available. The three segments dedicated to the kernel are:

- *kseg0*: When the three of MSB (most significant bit) of the virtual address are "100", the virtual address space selected is a 512-Mbyte kernel physical space. The processor will direct-map to physical address space. These references use cache memory, but they do not use TLB entries. Kseg0 is typically used for kernel executable code and kernel data.

- *kseg1*: When the three of MSB virtual address are "101", the virtual address space selected is a 512-Mbyte kernel physical address. The processor directly maps kseg1 to physical address space and do not use TLB entries unlike $kseg0$ and $kseg1$ uses uncached references. $Kseg1$ is typically used for I/O registers, ROM code and disk buffers.

- *kseg2*:When the two of MSB virtual address are "11", the virtual address space selected is a 1024 Mbyte kernel virtual space. Like kuseg, kseg2 uses TLB entries to map virtual addresses to arbitrary physical ones, with or without caching which the *N* bit in TLB entry determines whether the reference will be cached. Kseg2 is typically used for stacks and per-process data that it must be remap on context switches, for user page tables and for some dynamically allocated data areas.

During virtual-to physical address translation, the processor compares the PID and the highest 20 bits (the VPN) of the virtual address to the contents of the TLB.The Fig.3.7 illustrates the TLB address translation process.

A virtual address matches a TLB entry when the virtual page number (VPN) field of the virtual address equals the VPN field of the entry, and either the Global (G) bit of the TLB entry is set, or the process identifier (PID) field of the virtual address (as held in the EntryHi register) matches the PID field of the TLB entry. While the Valid (V) bit of the entry must be set for a valid translation to take place, it is not involved in the determination of a matching TLB entry. If a TLB entry matches, the physical address and access control bits (N, D and V) are retrieved from the matching TLB entry.

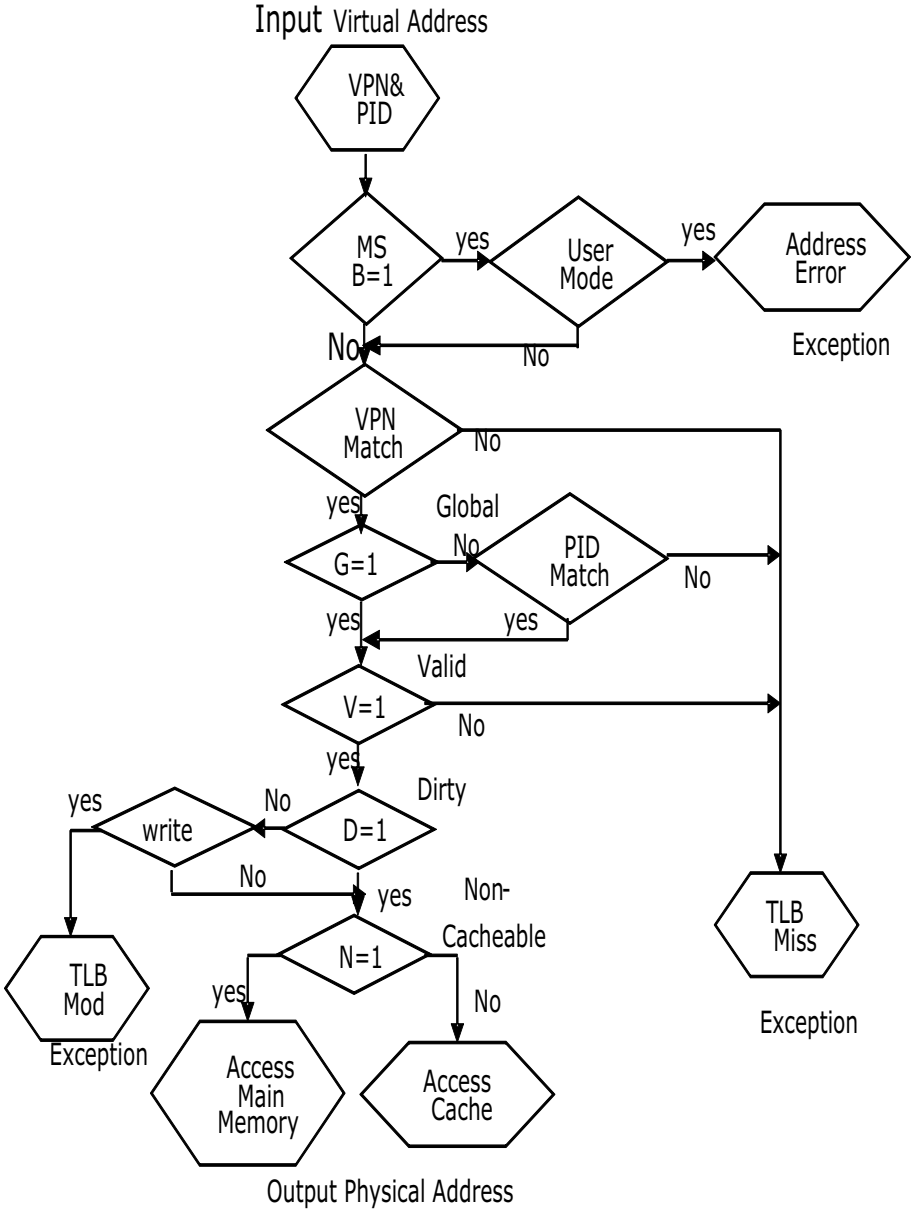


Figure 3.7: Address Translation Flow

Otherwise, a TLB miss exception occurs. If the access control bits (D and V) indicate that the access is not valid, a TLB modification or TLB miss exception occurs. If the N bit is set, the physical address that is retrieved is used to access main memory, bypassing the cache.

3.3.4.5 TLB Instructions in MIPS R3000

TLB must provide the instructions for the OS to read and write TLB entries. And the *EntryHI*, *EntryLO*, *index register*, *random register* are maintained by coprocessor 0 (CP0) which is called the System Control Coprocessor, is implemented as an integral part of the processor. CP0 supports address translation, exception handling, and other "privileged" operations.

- *tlbp* : It probes the TLB to see if an entry matches the *EntryHI* register contents. If a match exists, the processor loads the index(the index of the entry table) into the *index* register. Otherwise, it will sets the high order bit of the *index* register to 1.
- *tlbr* : It loads the *EntryHI* and *EntryLO* registers with the contents of the TLB entry specified by the contents of the *index* register.
- *tlbwi*: It loads the specified TLB entry with the contents of the *EntryHI* and *EntryLO* registers.The TLB entry is specified by the *index* register.
- *tlbwr*: It loads a pseudo-randomly specified TLB entry with the contents of the *EntryHI* and *EntryLO* registers. The TLB entry is specified by the contents of *random* register.

Also inside TLB we give out the exception tags for CP0, by using these tags the OS can choose the scheme to handle the exceptions.

3.3.5 Register Bank

The register bank is implemented using RAM blocks rather than explicit flip-flops. RAM blocks are single ported so two identical copies are required to provide the two simultaneous register reads. Despite this, they give a more compact solution than explicit registers. Read and write access is achieved by dividing each clock cycle into two phases. In the write back half of a cycle the register address to write to is sent to both RAM blocks. The data is written on the clock edge. To ensure data is never written to register zero the write enable line is de-selected if the write address is zero. The read portion of the cycle selects the address to be supplied to the RAM blocks to be the register addresses from the current instruction. These are then read for the next half cycle. After reading, the data is latched outside the register bank. The RAM blocks are explicitly preset to zero so R0 is never written to and will always be equal to zero.

3.3.6 Branch

Each cycle the PC updates to one of three values. The first possible value is PC+4. This is the most common case and the PC+4 value is generated using an incrementer. The second possibility is a JR instruction. This feeds a new number to the PC from a general register. It is important to pass the number from the forwarding paths if a newer valid value exists rather than directly from the register bank as the value there might not be updated yet. The third option is a branch, to execute a conditional branch firstly the conditions must be met. The two test registers are taken from the forwarding paths and compared if equal. This result and the sign of the first register are passed to the branch logic. The branch logic returns a flag to the multiplexer stating whether to load the branch target or the PC+4 value. The branch target is calculated by adding the current PC to the 16 bit sign extended immediate. There is one more case that is not covered and that is the jump. The jump takes a 26 bit immediate and places it in the bottom of the PC preserving the top four bits. This encoding was done by reusing the branch adder to save space. The bottom 26 bits from the PC are nulled leaving only the top four bits from the old PC value to be passed to the adder. The immediate is passed as a 26 bit rather than a 16 bit value. To record the return address the PC+4 value is latched and then multiplexed onto the pipeline as the result of the EX-stage.

3.3.7 Forwarding

The destination register number is usually explicitly specified in the instruction. Instructions that branch and link the return address need to have the destination register set to R31. All other instructions that do not have a destination have the destination register number set to 0. Each instruction unconditionally fetches the two registers specified in the instruction from the register bank. The destination register number is passed down with the instruction along the pipeline until it reaches the WB-stage. The requested source register number is compared to each of the destination register numbers flowing through the pipeline. If any of them match then the register bank value is discarded and instead the value from the forwarding path is multiplexed in. It is important that the value forwarding from the EX-stage takes priority over the one from the MEM-stage so as to get the more recent value. The comparators that detect if register numbers are equal also make sure that R0 is never forwarded as any calculated value is meant to be discarded and may be not equal to zero.

3.4 Leakage Power Analysis on MIPS R3000 Processors

Based on the structure presented in the last section, we have implemented a MIPS R3000 processor with Fujitsu's 65nm low-leakage CMOS technology. Table3.1 shows configuration parameters of the implemented processor. The implemented processor follows the specification of MIPS R3000 ISA; it has a standard 5-stage pipeline structure; 8K 2-way instruction cache and data cache have been integrated on-chip; and it has a 16-entry instruction TLB and a 16-entry data TLB.

Table 3.1: Processor Configuration

Parameters	
Processor Type	MIPS R3000
Target Frequency	200MHz
Instruction Execution	In-order
Pipeline Depth	5-stage
Cache Size	8K instruction, 8K data
Cache Type	Virtual-index, physical-tag
Replace Policy	LRU
Write Policy	Write Back
TLB Size	16-entry
TLB Type	Fully associative

As shown in Table3.2, the implemented processor has been described with Verilog HDL, and Synthesized with Snopsys's Design Compiler [69]. Synopsys's Layout tool Astro has been selected for the backend design, and Mentor's Calibre is used for verification [70]. The Back Side Bus (BSB) frequency is a half of the Front Side Bus (FSB) one; post-layout simulation shows the maximum operating frequency of the processor core is as much as 200MHz.

Table 3.2: Processor Implementation

Implementation Environment	
Process Library	Fujitsu 65nm CMOS process CS202SZ
Hardware Description	VerilogHDL
RTL Simulator	Cadence NC-Verilog 8.1
Synthesis Tool	Synopsys DesignCompiler 2007.12-SP3
P/R Tool	Synopsys Astro 2007.03-SP7
Verification	Mentor Calibre 2008.03
Die Size	4.2mm $\times$ 2.1mm

Leakage power analysis results have been obtained from post-layout simulation with Synopsys's Power Compiler [69]. We select four programs from MiBench [71] – Qsort, JPEG, Dijk-

stra,and Blowfish – as the test bench. The total leakage power consumption of the whole processor is 1029.2μW. Fig.3.8 shows the leakage share of each on-chip component.

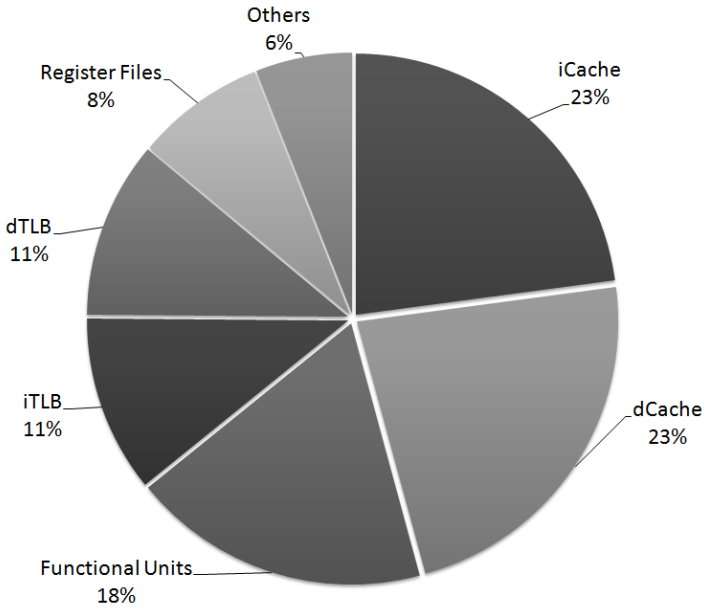


Figure 3.8: Leakage power analysis

As shown in the figure, the largest leakage consumer of the processor is the instruction cache and data cache, which take up 23% of the total leakage power consumption respectively. The high leakage share of caches comes from their large semiconductor footprint. Since cache lines are accessed partially, and their accesses show a strong regularity, the leakage-reduction mechanisms on caches have been well studied, as shown in Section 2.4.

The second largest leakage consumer is the functional units, which occupy about 18% of the total leakage consumption. In Chapter 4, the leakage-reduction mechanism on functional units will be presented.

Another big leakage consumer is the TLBs. The leakage share of instruction TLB (iTLB) and data TLB (dTLB) are 11% respectively. We will discuss the leakage-reduction mechanisms on them in Chapter 5 and Chapter 6.

The register files takes up another 8% of the total leakage consumption of the processor; and other on-chip components, such as pipeline latches, account for another 6%. Leakage-reduction mechanisms on them are out of the range of the thesis.

3.5 Conclusion

To reduce the leakage power of a embedded processor, we need to analyze the leakage share of each on-chip component. In this chapter, such an analysis is presented based on a MIPS R3000 processor. Here, the MIPS R3000 is selected because it has a very simple pipeline structure and it includes all essential components that can be found in modern embedded processors. Based on the analysis results, we pick up three components – functional units, instruction TLB, and data TLB as the leakage reduction target of this study. All of them take up a significant portion of the leakage power of an embedded processor, but have been well studied because of their high utilization. The corresponding leakage reduction mechanisms on these selected target will be presented in the following chapters.

Chapter 4

Fine-grained Power Gating on Functional Units

4.1 Introduction

Power Gating (PG) is one of the most effective leakage-reduction techniques, with which circuit blocks are not connected directly to the power grid but through power switches. To reduce the leakage power, the connection between idle circuit blocks and their power supply can be temporarily cut off by turning off power switches. In recent commercial microprocessors, core-level PG [72] has been implemented by inserting power switches between the global power grid and the power ring of processor cores. When the operating system knows the idle state of a processor core may last for certain times, the core, which can be either the CPU-core [73] or other heterogeneous cores [74], will be put into the sleep mode by shutting off its power supply. Although such a core-based PG control scheme is straightforward and easy to be applied, it misses leakage-saving opportunities when a portion of intra-core components are in the idle state. Moreover, the wakeup latency of core-level PG, which is the time needed to fully restore the power of a sleep core, is in the order of micro-second. That implies, with core-level PG, only conservative PG control schemes can be applied when a long idle time of the processor core is detected.

In contrast, PG control schemes which aggressively power on/off functional units within a processor core have also been studied [75] [76] [77]. By exploiting PG opportunities at a finer granularity, these schemes usually achieve better leakage reduction effects than the core-based PG scheme. However, PG is a non-ideal technique, and aggressive power-on/off functional units may incur unaffordable penalties on both performance and power consumption. Hu et al. [75] analyze the costs involved in power-gating functional units and present an analytical model of the Break-Even Time (BET), which is the minimum time a functional unit should remain in sleep mode such that the saved leakage energy can compensate the dynamic energy overhead caused by powering on/off the unit. If a sleep event is shorter than BET, PG consumes more power instead of saving. To avoid such short-term sleep events, in the same paper, they have proposed a time-based PG control policy and a branch-guided policy. Youssef et al. [76] have further exploited PG opportunities by tracking the executed program behavior across different time segments and predicting the length of idle periods

of functional units; and Lungu et al. [77] have proposed a scheme to guarantee the quality of PG with a successful monitor. In addition, compiler-based schemes [78] [79] [80], which employ static code analysis or dynamic profiling to identify the time period when a functional unit is not used, have also been proposed.

However, all above papers miss the corresponding circuit-level techniques. Since functional units are power-gated in response to the workload at runtime, high-speed fine-grained PG techniques that can better exploit leakage-saving opportunities spatially and temporally are indispensable.

In our previous works [81] [82] [83] [84], we have presented a framework to implement the fine-grained PG on microprocessor functional units by integrating circuit-level, architecture-level, and system software techniques. At circuit-level, we have proposed a fine-grained PG technique, which has nano-second order wakeup latency and can be implemented at arbitrary granularity. At architecture-level, a PG control scheme, which keeps a functional unit active only when being used, has been applied. In addition, BET-aware PG control schemes, which are guided by the system software (compiler and operating system) have also been proposed to achieve maximum leakage reduction effects.

In this chapter, we apply the proposed framework to a real-chip implementation. The contributions of the paper can be summarized as follows:

- We have proposed a fully automated design flow of fine-grained PG, which has nano-second wakeup latency and can be implemented at arbitrary granularity.
- We have implemented a MIPS R3000 prototype chip with Fujitsu's 65nm CMOS technology. The prototype chip integrates the fine-grained PG technique into its functional units, which can be power-gated individually at runtime. To the best of our knowledge, this is the first chip which provides fine-grained PG control schemes on functional units.
- Unlike previous works which evaluate the leakage-saving effects with post-layout simulations, comprehensive real-chip evaluations have been executed to measure leakage-saving results and other important parameters (BET, wakeup latency and so on). We have measured these parameters at different temperatures by using a thermal chamber, and the obtained results are more reliable than those obtained from simulations.

The rest of this chapter is organized as follows. Section 4.2 introduces the fine-grained PG design methodology; and the PG control schemes on functional units are presented in Section 4.3. In Section 4.4, we illustrate the implementation of the prototype chip; and the real-chip evaluation results are shown in Section 4.5. Section 4.6 is conclusion and the future work.

4.2 The Fine-grained PG Methodology

In this section, we illustrate the fine-grained PG technique which is integrated into the functional units design in this work to reduce leakage power at runtime. In addition, a fully automated design flow will be presented.

4.2.1 Fine-grained PG

Unlike the conventional PG, fine-grained PG used this work requires a set of special cells each of which has its own virtual ground (VGND) line (such a set of cells is referred as PG-cells in this thesis). As shown in Fig.4.1, VGND lines from several cells are connected through a shared high-Vth footer power switch to the real GND. If the sleep signal is set on, the logic block is put into the sleep mode by cutting off the connection between VGND and GND. In this case, the VGND is charged up to a voltage near the VDD, and the leakage current is reduced consequently. When the power switch is turned on, parasitic capacitance on the VGND line is discharged through the power switch, and a certain amount of time (wakeup latency) is required to wait the voltage on VGND to become stable.

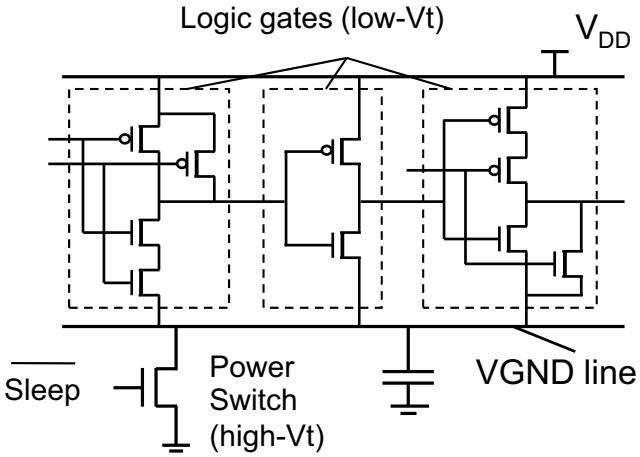


Figure 4.1: Out line of power gating circuit

The wakeup latency is affected by physical parameters such as power switch size and VGND capacitance. In order to fine tune these parameters, we used a Locally-Shared Virtual ground (LSV) scheme [85] shown in Fig.4.2. With this scheme, the entire PG target is partitioned into smaller local power domains, and the VGND line and the power switch are shared only within a local power domain. Although power switches are controlled by a unique control signal for each PG target, the size of power switches can be tuned independently, implying that the IR-drop on the VGND line can be managed easily by selecting appropriate-sized power switches for each local power domain. Furthermore, since existing ground rail in the PG-cells is used as the real ground, permanent power networks are reachable throughout the PG target; thus, non-power-gated cells such as flip-flops,

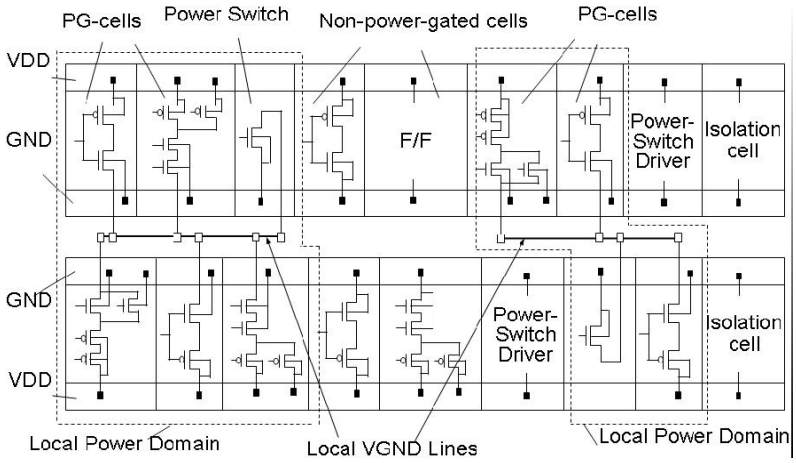


Figure 4.2: A VGND architecture for a fine grain PG

clock buffers, repeaters, power switch drivers, and isolation cells can be distributed arbitrarily among local power domains without incurring power-routing. Furthermore, an in-rush current suppression mechanism [86], which skews the wakeup timing of each local domain by simply down-sizing a portion of the leaf drives of the power-switch-driver tree, has also been adopted with the LSV scheme.

Compared with the UPF-based methodology [87], the LSV scheme can control the size and the number of local power domains by taking into account the given requirement on wakeup latency. As a result, the wakeup latency of fine-grained PG is typically less than a few nano-seconds (we will confirm this in Section 4.5). Moreover, there is no constraints on the placement of power switches and non-power-gated cells as the UPF-based methodology, and the power integrity issues caused by power-routing can be minimized. Compared with the cell-based PG methodology [88], which integrated a power switch in each primitive cell, our scheme has less area overheads. Compared with the ring-based PG [88], the IR-drop target can be managed easily with smaller number power switches, and the non-power-gated cells can be placed within a PG target with our methodology.

4.2.2 The Design Flow of Fine-grained PG

To facilitate the design process with fine-grained PG, we newly developed a fully automated design flow. As shown in Fig.4.3, the design flow can be explained as follows:

1) A set of PG-cells is generated, generated cells include GDS file and the timing library.

2) An RTL model of a PG target with sleep-control signals is designed.

3) The RTL model is synthesized by using Synopsys Design Compiler.

3) Isolation cells are inserted to all the output ports of the synthesized netlist in order to prevent the propagation of floating output values when the PG target is in the sleep mode.

4) The netlist with the isolation cells is placed by using Synopsys Astro.

5) The local power domain is partitioned, local VGND lines are formed, and the power switches are inserted between the VGND and GND lines by using Sequence Design's CoolPower [89].

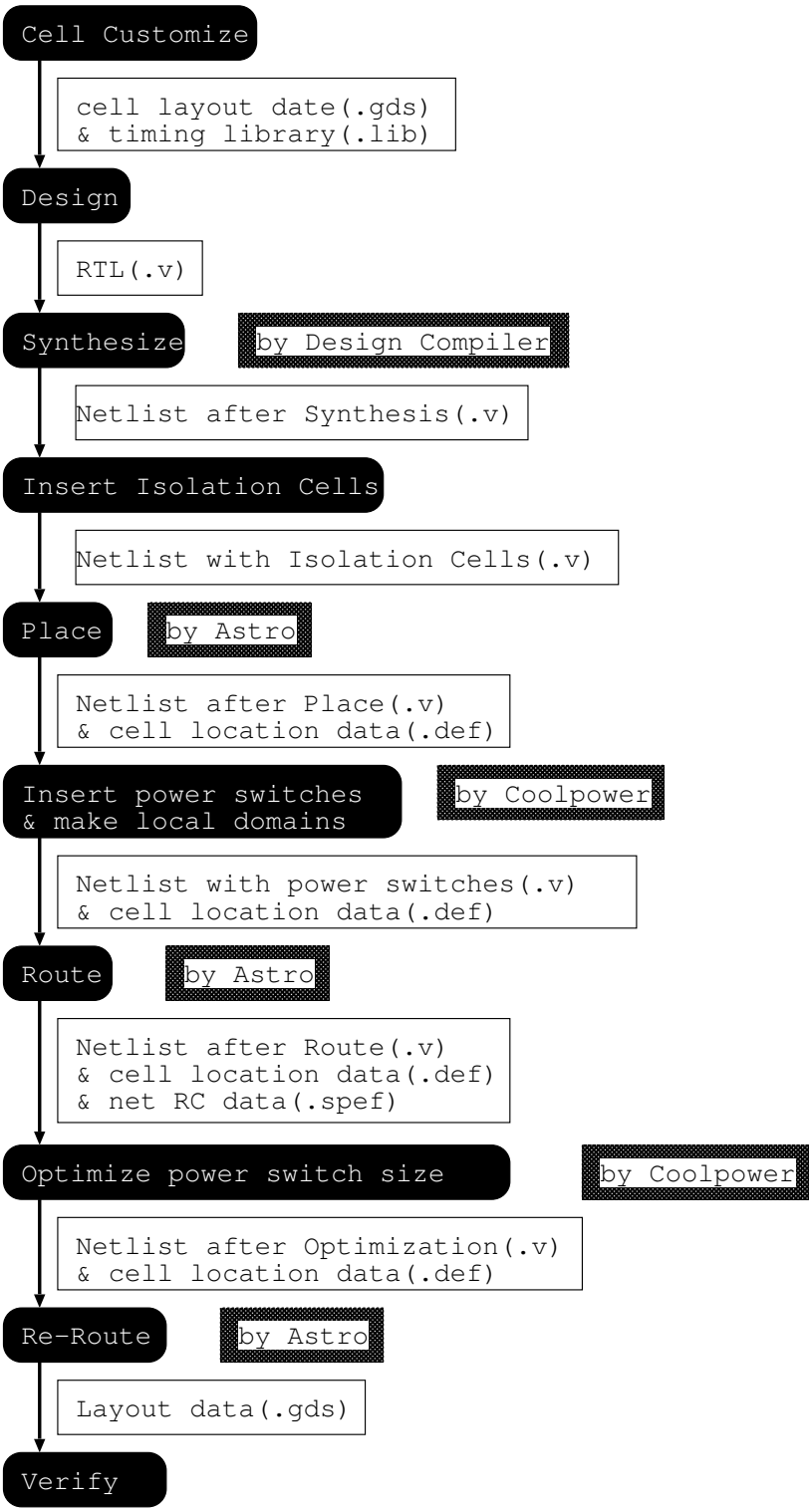


Figure 4.3: Design Flow

6) The netlist with the power switches is routed by using Synopsys Astro.

7) The previous two steps are performed again for the purpose of VGND optimization, power switch sizing, and routing.

In the end of the design flow, the GDS file of a PG target will be generated. Since PG-cells can coexist with common cells in a row, the conventional timing-driven placement and routing methodology can be used. Furthermore, this flow is fully automated, and the additional design complexity for the fine-grained PG is small.

4.3 Runtime PG Control Schemes on Functional Units

In this section, we propose PG control schemes which can dynamically power on/off functional units in response to workloads of programs currently running on the processor. Here, a widely used 32-bit embedded processor, MIPS R3000 [8], is selected as the target processor. As shown in Fig.4.4, MIPS R3000 provides a standard five-stage pipeline structure, consisting of Instruction Fetch (IF), Instruction Decode (ID), Execution (EX), Memory Access(MEM), and Write Back (WB). To apply the fine-grained PG technique to MIPS R3000, we select following units as the PG targets:

- CLU (Common arithmetic and Logic Unit)
  A general computational unit for addition and subtraction operations. It can be put into the sleep mode when branch, NOP, or memory access instructions without address calculation are fetched.

- Shifter
  A barrel shifter which can shift a data word by a specified number of bits in one clock cycle. Since the shifter occupies the considerable area but is not so frequently used, it is implemented as an individual unit.

- Multiplier
  A 32-bit multiplier which takes 4 clock cycles for each multiplication operation. If upper 16-bit of either operand is all-0, the upper part of the multiplier can be put into the sleep mode.

- Divider
  A 32-bit Divider which takes 10 clock cycles for each Division operation.

Note that, functional units occupy about 55% of the area of the processor core (without on-chip caches and TLBs), and their usage is easy to be identified based on the fetched instructions. Moreover, all of them are implemented with combinational circuits, implying that state retention techniques are not required. Thus, only 1-bit sleep signal is needed to control the mode of each unit.

4.3.1 Fundamental PG Control Policy

A fundamental PG control policy tries to put each functional unit into the sleep mode right after finishing its operation. As shown in Fig.4.4, the mode of function units is controlled by sleep signals that are generated from a dedicated sleep controller. When an instruction is fetched in the IF stage, the sleep controller checks the fetched instruction and judges which unit is to be used. Here, the target working frequency of the processor is set to 100MHz, and we assume the wakeup latency is 1 clock cycle (10ns). To hide the wakeup latency, a simple decoder is provided in the IF stage to detect which functional unit will be used by the currently fetched instruction. As soon as a functional unit is detected, a sleep signal will be generated and sent to the unit immediately. Thus, when

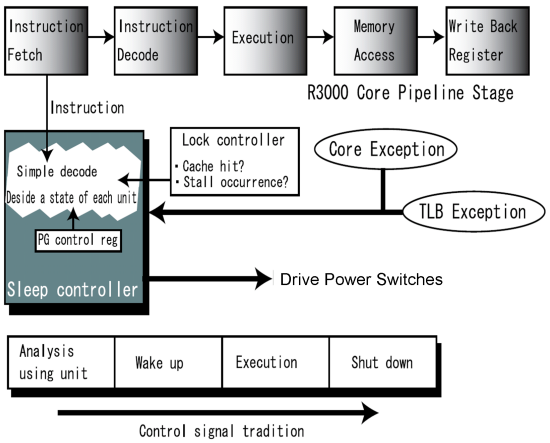


Figure 4.4: Sleep control by the fetched instruction

the instruction reaches the EX stage, the required unit had already been fully powered up, and no performance detriment will be introduced by the fundamental policy.

When an instruction is fetched in the IF stage, the decoder checks the upper most 6bits of the instruction, and judges whether the instruction executes a R-Type operation (ROP) [8]. If so, the functional unit to be used can be identified by the last 6bits of the instruction. Otherwise, extra judgments are needed to decide whether it is an I-Type instruction which uses CLU for address-calculation.

4.3.2 BET-aware PG Control

PG is not a non-overhead technique. As shown in Fig.4.5, the state transfer between the sleep mode and the active mode consumes extra dynamic power for power switches, isolation cells and buffers in sleep signal wires. Even in the sleeping mode, leakage current decreases with time but still continues to flow until the capacitance on the VGND and output nodes of logic gates are charged up. The Break-Even Time (BET), which means the time point when the aggregated leakage energy savings equal to the energy overheads due to the power domain state transfer, is a crucial parameter to assure the overall power saving effect.

The leakage reduction effects of the fundamental PG control policy are sensitive to the BET in that functional units are switched between the sleep mode and the active mode frequently. Since extra power consumption is induced by powering on/off functional units as well as the sleep controller, the fundamental policy has a risk to increase the power consumption instead of saving. For example, when multiplication operations are executed iteratively with a small interval, the multiplier will be woken up soon after its shut-off. If the sleep time of multiplier is less than the BET, the mode-transition overheads of PG will increases the total power consumption. In this work, we employ the compiler to detect instructions that may cause small sleep intervals (smaller than BET); and mode-transition overheads can be eliminated by keeping functional units active after its operation. For this

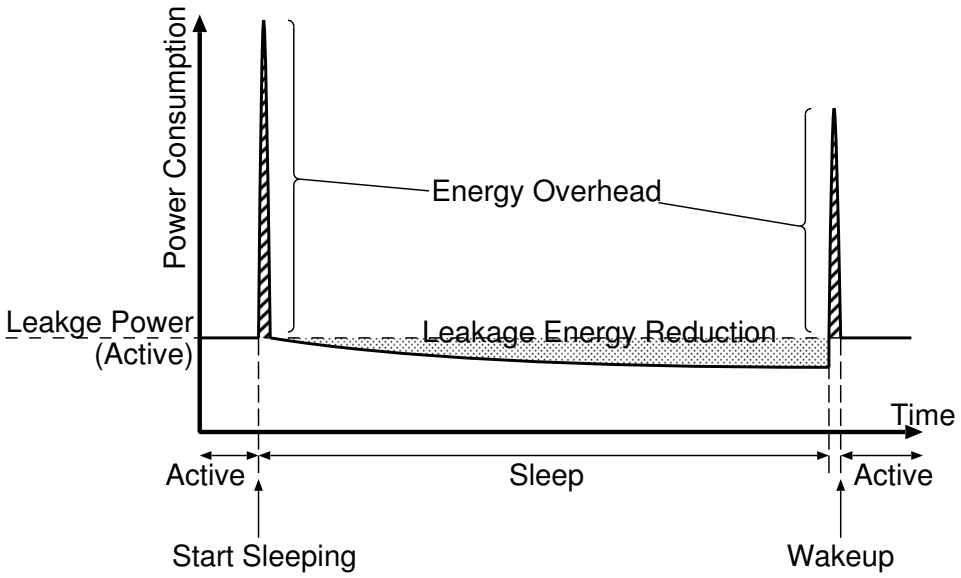


Figure 4.5: Power Consumption on Mode Transfers

purpose, we introduce a set of non-PG instructions.

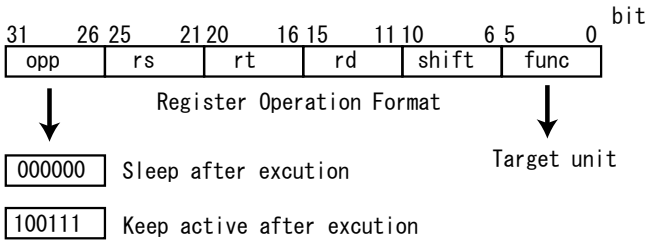


Figure 4.6: Non-PG Instructions

Fig.4.6 shows an example of non-PG instructions. In MIPS R3000 ISA, when the most upper 6bits (ope-code) is all-0, the instruction performs computational operation; and the type of operation is determined by the last 6bits. Here, we used "100111", which is not defined in the original ISA, as the upper 6bits to indicate non-PG instructions. After executing a non-PG instruction, the corresponding functional unit will not be power-gated, but kept in the active state. Such an active state will be kept until the another instruction, which use the same functional unit but with all-0 ope-code, is executed. Thus, by replacing small-interval instructions with non-PG instructions, the overhead of power gating can be avoided.

Furthermore, leakage power is sensitive to the temperature, that is, it increases exponentially as the temperature rises. BET is also influenced by the temperature but in the opposite direction – BET becomes shorter at higher temperatures (we confirmed this in Section 4.5). Such a characteristic can be exploited to achieve better leakage reduction effects. When the chip is working at a low temperature, PG policies that power on/off functional units conservatively should be used to avoid mode-transition overheads; while more aggressive policies should be adopted in higher temperatures.

In this work, we have proposed three different PG control policies that can be changed dynamically according to the chip temperature. The three policies include: (1) fundamental, runtime PG policy (as mentioned in subsection 4.3.1), (2) units going to the sleep mode only when a cache miss happens, and (3) units never going to the sleep mode. These policies are applied based on the value stored in a PG policy register, which can be written only in the kernel mode; and the operating system decides which policy should be used according to the information from an on-chip leakage monitor.

4.4 Geyser-1 Prototype Chip

To demonstrate the feasibility of the fine-grained PG technique and prove the leakage reduction effects of the PG control scheme presented in section 4.3, a prototype chip, Geyser-1, has been implemented.

4.4.1 Design Policy

Fine-grained PG complicates the power grid design during layout. For this reason, our first prototype chip, Geyser-0 [81] [82], failed to work due to unexpected power-rail shorts. To simplify the back-end design, the second prototype chip, Geyser-1, has been designed with following policies: (1) only the CPU core is implemented on a chip. Caches and TLBs provided in Geyser-0 are moved off-chip. (2) The design flow is improved so that no manual edit on the layout is needed. (3) 65nm Fujitsu's high-Vth CMOS process is used instead of 90nm standard process used in Geyser-0.

The decision to move caches off chip has serious impacts on the back-end design. Because of the pin-limitation problem, a part of address/data signals must be multiplexed. Additional delay of such multiplexers, long wires and I/O buffer to access off-chip caches severely degrades the operating frequency. Moreover, the package technique used for Geyser-1 also imposes limitations on the maximum frequency. As a result, the maximum clock frequency of Geyser-1 is set to be 60MHz at the layout stage.

4.4.2 Implementation

Geyser-1 has been described by Verilog HDL, synthesized with Synopsys Design Compiler 2007.03-SP4, and layouted by using Synopsys Astro 2007.03-SP7. Fujitsu's 65nm 12-metal-layout CMOS library CS202SZ (high-Vth process) is used as the standard cells library, in which core cells work at 1.2V while I/O cells working at 3.3V. As illustrated in Section 4.2, the fine-grained PG technique requires a set of customized PG-cells. We selected 106 cells from Fujitsu CS202SZ cell library and modified them to have separate VGND lines. These cells are used to build functional units during the placement and routing phase; and other parts of the processor use the common cells. In addition, we has designed power switches and isolation cells, which are also required by the fine-grained PG design flow.

Power switches are inserted in the post layout netlist by using Sequence Design's Cool Power 2007.3.8.5. Since the inserted power switches will increase the voltage of VGND (IR-drop problem), the performance of functional units in the active mode may be degraded.

As mentioned above, the critical path of Geyser-1 sits in the IF/MEM stage, where the off-chip cache-access happens. Thus, by appropriately setting the IR-drop target according to the timing slack existing in the EX stage, the cycle time degradation of the whole processor can be avoided. Here, we determine the IR-drop target based on the timing analysis of each functional unit (With the proposed

design flow, the IR-drop of a PG target can be managed easily, and we can set the IR-drop target of each functional unit independently). As a result, for the multi-cycle multiplier and divider, which have a large amount of timing slack, the IR-drop target is set as 200mV; while for CLU and shifter, whose timing slack is tighter, the IR-drop is set as 100mV. In both cases, no performance degradation will be incurred.

Table 4.1: The area overhead of fine-grained PG

	Total(μm^2)	PS(μm^2)	ISO(μm^2)	Overhead
CLU	3752.8	296.4	79.2	10.3%
SHIFT	3078.0	298.8	76.8	12.6%
MULT	23863.6	1762.0	153.6	8.5%
DIV	27918.4	1301.2	153.6	5.4%
others	46304.4	-	-	-

Table4.1 shows the area of each functional unit. In the table, PS means the area of power switches, while ISO stands for the area of isolation cells. The overhead of the fine-grained PG is 5.4% - 12.6%, which are mainly caused by power switches and isolation cells.

The cells introduced by fine-grained PG (e.g. isolation cells and PG control circuit) will also cause additional leakage power consumption. According to the circuit-level simulation, the leakage overheads caused by isolation cells and PG control circuit (including the pre-decoder in the IF stage) are 0.44% and 3.03% of the whole processor.

Fig.4.7 shows the layout of Geyser-1. The chip size is 2.1mm ×4.2mm. As shown in the figure, the four black boxes in the middle of the layout are functional units which are implemented with fine-grained PG; and four small black boxes located near corners are leakage monitors.

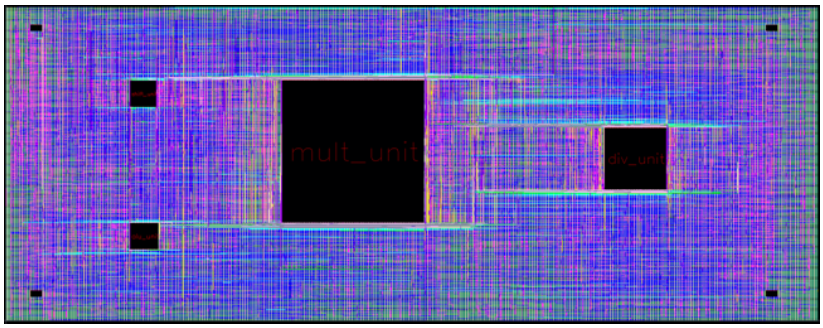


Figure 4.7: Layout of Geyser-1

4.5 Real-chip Evaluations

In order to evaluate the chip, we developed a dedicated board including an Virtex-4/LX FPGA board and socket for Geyser-1 chip. The power supply of Geyser-1 chip is completely separated from others for the purpose of accurate current-measurement. Fig.4.8(a) and Fig.4.8(b) show our test environment. A thermal chamber is utilized to heat the whole system, and measure the power consumption of the chip at different temperatures. Since caches and TLBs are not included in Geyser-1, small and high speed memory for storing instructions and data are implemented with BRAMs in the FPGA.

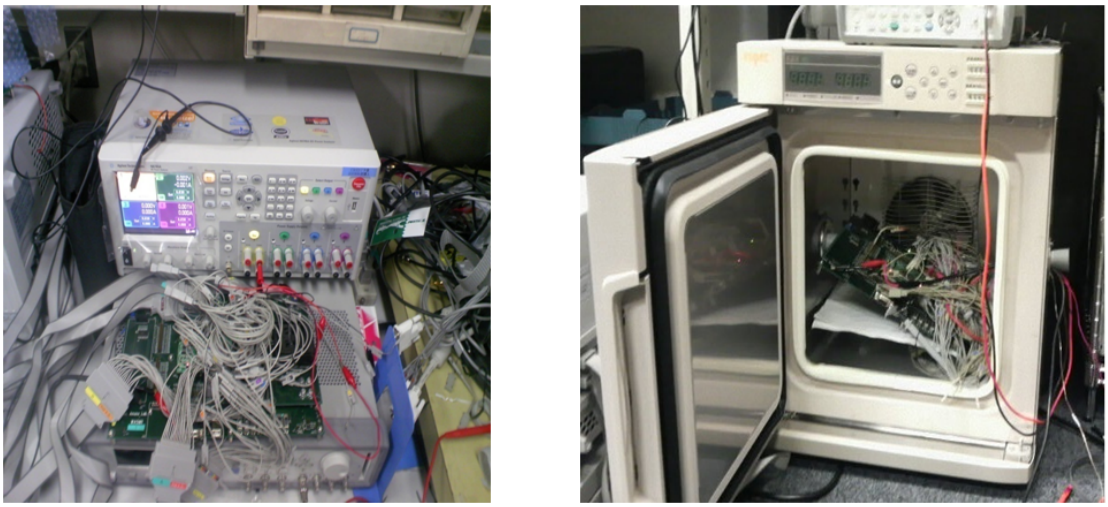


Figure 4.8: Test Environment

4.5.1 Clock Frequency and Wakeup Latency

First, we evaluate the maximum clock frequency and the wakeup latency. For this purpose, we define two different working mode of the chip. The RTPG mode means the test program running on the processor does not include any non-PG instructions, and a functional units will be immediately powered off after its operation. On the other hand, the ACT mode indicates that all functional instructions are replaced by non-PG instructions. When working at such a mode, functional units will always stay in the active mode.

A simple benchmark program is executed. When working at 60MHz, the prototype chip works correctly at both the ACT mode and the RTPG mode. Since we assumed the wakeup latency is one clock cycle (Section 4.3), the evaluation result proves that the wakeup latency of the fine-grained PG is less than 17ns.

4.5.2 BET

Since BET is an important design factor of fine-grained PG, we evaluate BET of each functional unit on the real chip. Here, we take the multiplier as an example to illustrate our measurement strategy. The test program is a simple loop which consists of a multiplication instruction, several idle cycles, and a return to the loop entrance. The longer the idle interval is, the less frequently the mode-transition happens; hence, the better power-saving can be achieved by PG. With such a measurement strategy, we investigate BET by changing the length of idle intervals and comparing the power difference between RTPG mode and ACT mode. Fig.4.9 shows obtained values of the multiplier. The horizontal axis presents the length of idle interval in the form of clock cycles; while the vertical axis is the power difference of the test program which is executed at ACT mode and RTPG mode respectively. The BET is the interval at which the power difference of ACT mode and RTPG mode becomes zero. As shown in Fig.4.9, when working at 50MHz and 25℃, the BET of the multiplier is 44 clock cycles. Table4.2 presents the BET of four functional units when working at 25℃, 65℃, and 100℃, respectively. As shown in the Table, BET decreases exponentially as temperature increase. Note that, with BET-aware PG control schemes mentioned in section 4.3, the compiler will use the non-PG instructions if the interval of two instructions, which use the same functional unit, is less than the BET.

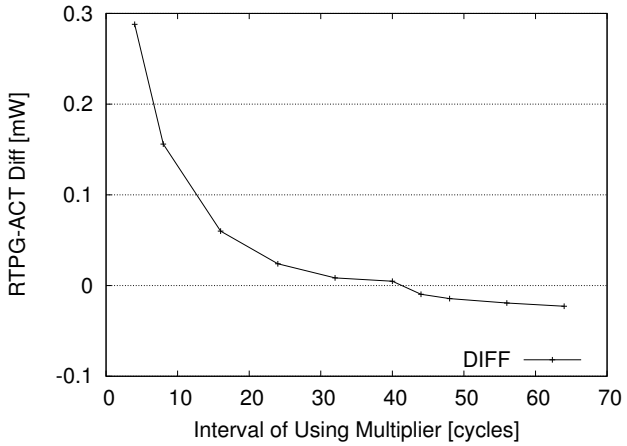


Figure 4.9: BET of Multiplier

4.5.3 Leakage Power Reduction

In this subsection, we measure the leakage power when all functional units are staying in the active mode and the sleep mode respectively. After putting functional units into a given mode (sleep or active), we stop the clock signal, thus no dynamic but only leakage power of the processor can be measured. As shown in Fig.4.10, the PG can reduce the leakage power of the whole processor core by 5% at 25℃. It is worth noting that, when the temperature grows up, the effect of PG becomes

more obvious.

Table 4.2: BET of functional unis (clock cycles)

	25℃	65℃	100℃
CLU	54	14	6
SHIFT	67	16	7
MULT	44	12	5
DIV	32	9	4

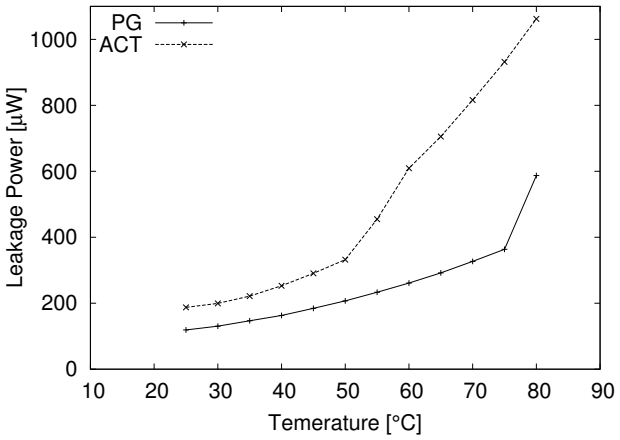


Figure 4.10: Leakage power reduction of Geyser-1

4.5.4 Evaluations with Benchmark Programs

Evaluations with benchmark programs are also executed. We select two programs from MiBench [71]: Quick Sort (QSORT) from mathematics package and Dijkstra from the network package. In addition, we also use DCT (Discrete Cosine Transform) from JPEG encoder program as an example of media processing. Since the delay of the Block RAM inside the FPGA is large, the evaluation is performed at 10MHz.

Fig.4.11, Fig.4.12 and Fig.4.13 show the power consumption of three benchmark programs working with PG and without PG. The best power reduction effect is achieved in Dijkstra which does not use the multiplier and divider. The total power consumption of the processor core can be reduced by 8% at 25℃ and 24% at 80℃. Note that, power reduction effects are better than the values shown in Fig.4.10. It is not strange because by putting functional units into the sleep mode, redundant dynamic power consumptions, which caused by incomplete operand isolation, can also be eliminated. The power reduction of Quick Sort, which occasionally uses the multiplier and divider, is from 4% to 29%; and for DCT, which uses multiplier frequently, the power is reduced by 3% at 25℃ and 17% at 80℃.

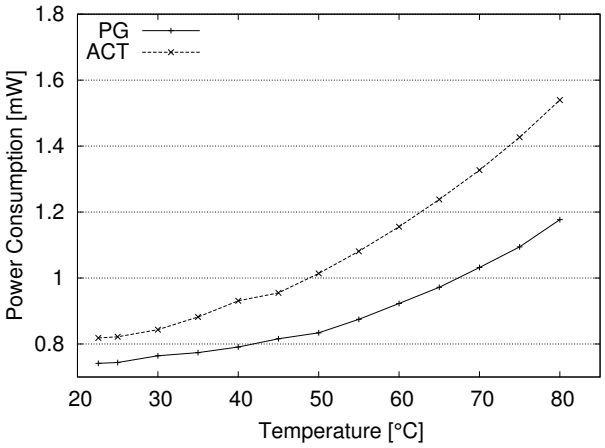


Figure 4.11: Power for Dijkstra

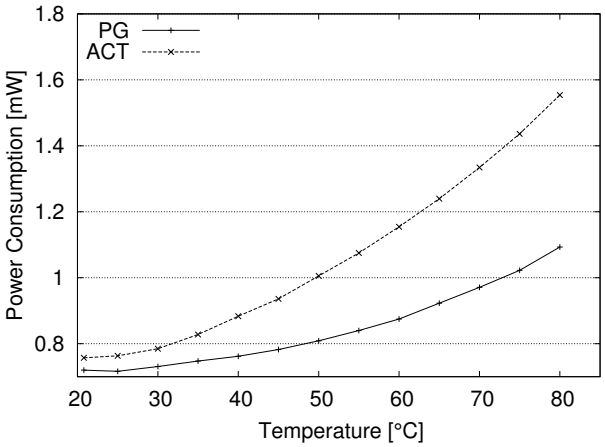


Figure 4.12: Power for QSORT

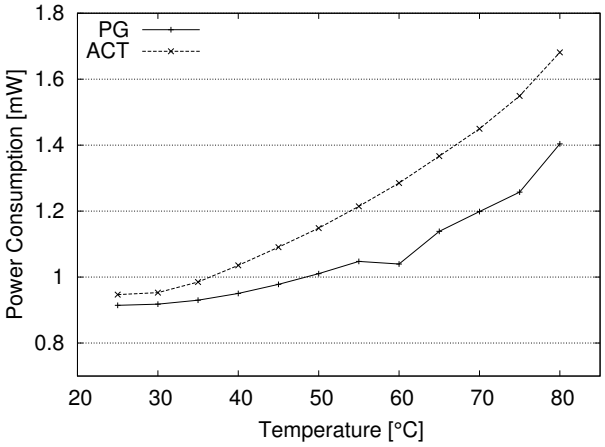


Figure 4.13: Power for DCT

4.6 Conclusion

Geyser-1, a prototype MIPS R3000 processor which implements the fine-grained power gating control on its functional units, has been presented. In this chapter, we have discussed the design methodology of fine-grained power gating technique, the power gating control policy on microprocessor functional units, and the real-chip implementation and evaluation. To simplify the design process with the fine-grained power gating technique, a fully automated design flow have been also proposed. The real-chip evaluation of Geyser-1 shows the fine-grained power gating can reduce the power consumption of the the processor core by 5% at 25℃ and 23% at 80℃.

Chapter 5

Leakage-efficient Instruction TLB

5.1 Introduction

The translation Look-aside Buffer (TLB) is an important component in embedded processors. It provides storage attributes, access permissions and virtual to physical address-translation to efficiently address huge amounts of physical memory. To avoid the performance degradation caused by TLB misses, modern embedded processors usually adopt large size TLBs with fully associative structure, which lead to a non-trivial energy dissipation of both dynamic and leakage. For example, in the MIPS R3000 processor presented in Chapter 3, a 16-entry TLB consumes as much as 11% of leakage and 19% of dynamic power of the whole power consumption of the processor. Many publications have been devoted to exploring the dynamic power reduction mechanisms on TLBs [90] [91] [92], either by reducing the energy dissipation per TLB access, or reducing total number of TLB accesses. However, as the leakage power has emerged as a limiting factor, power-optimized TLB design only addressing on dynamic power becomes insufficient.

Furthermore, TLBs are also one of the on-chip "hot-spots", due to the high power density. According to the simulation results from paper [92], the power density of an instruction TLB is 8 times higher than that of an instruction cache. Since the leakage power varies exponentially with temperature, the TLB is one of the most "leaky" components on a processor.

This chapter focuses on reducing the leakage power of the instruction TLB (iTLB). Although TLBs have a cache-like structure, blindly transplanting cache leakage reduction mechanisms, such as cache decay [47] and drowsy cache [48], into an iTLB design will introduce unacceptable overheads in terms of both performance and power consumption, due to their different access pattern, replacement policy and mis-recover penalty. Further, the iTLB is one of the most active components in embedded processors with a high utilization, which intuitively does not leave much space for leakage reduction. Fortunately, the page-based iTLB references exhibit strong locality, and when programs enter a physical page, following instructions tend to be fetched from the same page for a considerable long time. Hence, by inserting a small size storage component, which keeps the recent address-translation information, between the processor and the iTLB, a majority of address-

translation requests can be satisfied with the small component without accessing the iTLB. Then, with integration of the Dual Voltage Supply (DVS) technique, the iTLB can be put into low-leakage mode (with the lower voltage supply) and restored to the active mode only when the iTLB look-up becomes necessary. Based on such a design philosophy, a leakage efficient iTLB design is proposed with three different leakage control policies. Power evaluation results show the proposed design can reduce the iTLB leakage power by 50% with negligible performance degradation.

It is worth noting that although we focus on reducing leakage power of the iTLB, the proposed design can be easily integrated with the clock gating technique to reduce dynamic power as well. According to the power evaluation results, 75% of the dynamic power of iTLB can also be reduced. To the best of our knowledge, no previous work can provide such an uniform low power iTLB solution.

The remainder of this chapter is organized as follows. The next section presents a leakage efficient iTLB structure based on the locality analysis of iTLB references. In Section 5.3, detailed leakage control policies and hardware implementation will be illustrated. Leakage saving results will be shown in Section 5.4, and we will discuss the related work in Section 5.5. Section 5.6 concludes the chapter.

5.2 Design Philosophy

Low-leakage design can be classified into two categories: the one using static leakage control mechanisms and the one using dynamic leakage control mechanisms. The static leakage control mechanism trade increased circuit delay for reduced leakage power by selecting slower but lower leakage transistors (e.g. high-Vth transistors) at the design time. Since the iTLB is usually on the critical path, and using slower transistors on it will directly degrade the maximum frequency of a processor, in this work, we turn to the dynamic leakage control mechanism instead.

The dynamic leakage control mechanism achieves leakage saving by putting a design target into the low-leakage mode during the idle period, and its leakage reduction effects rely on the scale of the design target, the time duration in the low-leakage mode and the mode-transition frequency. Since the iTLB is one of the most active components in embedded processors with a high utilization, there does not seem to be much room left for dynamic leakage control either. In this work, we try to reduce the leakage power consumption of iTLBs by exploiting the locality hidden in the instruction stream from the perspective of the page-based iTLB referencing. The contents of this section are organized as follows. After a brief introduction on the experimental infrastructure engaged in this work, we will analyze the iTLB referencing locality and corresponding leakage reduction opportunities quantitatively. Then, based on the analysis results, a leakage efficient iTLB structure will be presented.

5.2.1 Experimental Setup

The locality analysis in this section is based on trace-driven simulations, where the trace data are obtained from the MIPS system emulator of QEMU [93]. To better emulate the interaction between the iTLB and the Operating System (OS), the emulator boots up a linux system (Debian in this work), on which eight application programs from different fields of MiBench [71] are executed. A group of authors [94] modified the basic structure of QEMU, so that it can be used to trace both TLB references and TLB-flush information. Table5.1 shows the configuration parameters of the emulator, and the number of TLB-flushes of each application program is shown in Table 5.2. Note that, when application programs are executed, several OS related processes are running on the background, and there are also some basic processes, like Shell, are running with application programs concurrently.

5.2.2 Locality Analysis

Generally, a TLB miss is handled as an exception, which incurs long mis-recovery penalties and may degrade the performance of a processor significantly. To reduce the TLB miss rate, the iTLB in modern embedded processors is usually organized as a fully associative structure, implying that all iTLB-entry should be accessed for every instruction-fetching. On the other hand, high locality consists in the instruction stream: instructions are fetched in program order, conditional jumps tend

Table 5.1: Configuration Parameters

Trace Environment	
CPU Type	MIPS R3000
Instruction Execution	In-order
OS Type	Debian
Kernel	Linux 2.6.15
Shell	ash
Compiler	GCC(4.2.2)

Table 5.2: TLB-flushes of Application Programs

Programs	TLB-flushes	Programs	TLB-flushes
BasicMath	72	Dijkstra	59
JPEG	73	Qsort	32
FFT	49	SHA	87
Susan	52	Rsynth	102

to jump close by, and loops repeat the same code multiple times. From the perspective of page-based iTLB referencing, where the page transition is mostly due to function calls/returns and long distance jumps, the locality of instructions can be translated into a same-page-hit behavior. Fig.5.1 shows the miss rate of an iTLB by varying the number of entries from 1 to 64. Here, the iTLB miss rate, which is a simple proxy of locality, is employed to better understand the iTLB referencing locality and page-transition behaviors. As shown in the figure, the high degree of iTLB-referencing locality is rather obvious as an overall low miss rate can be observed for all 8 application programs. Note that, even small size configurations can also achieve a quit low miss rate. For instance, the average 1-entry miss rate, which reflects the referencing locality and page-transition behaviors directly, is less than 2%. Although the simulation results are obtained from the typical embedded applications, the high locality of instruction fetching can also be observed from more generous applications. To prove this, the 1-entry iTLB miss rate has been evaluated by using SPEC2006 Integer Benchmarks [95]. The simulation is based on the ZESTO simulator [96], and for each program 50,000,000 instructions have been executed by skipping the first million. The evaluation results show that the average 1-entry miss rate of all 12 applications is lower than 5%, and only 2 programs' miss rate (perlbench and xalancbmk) is higher than 7%, but still less than 10%.

The above observation reveals the most important iTLB referencing characteristic that we employ in this work to fight leakage – when a program enter a physical page, the same-page instruction-fetching tends to sustain a long time. Thus, if same-page-hit iTLB references can be detected and treated differently, the frequency of iTLB accessing can be drastically reduced, which makes the iTLB itself an excellent target for leakage saving.

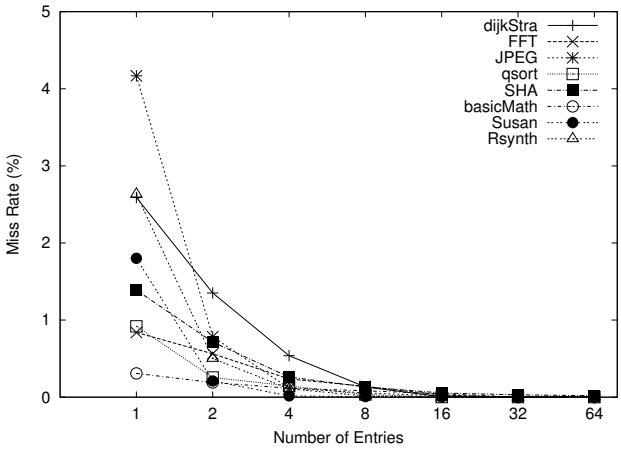


Figure 5.1: iTLB Miss Rate

Another perspective on iTLB referencing locality is from the variation of the miss rate among different configurations. Although the miss rate of iTLB continually decreases as the TLB-entry increasing from 1 to 64, entry-rise at the lower end of the x-axis has more significant miss reduction effects than at the higher end. For example, increasing the number of entry from 1 to 2 can reduce the miss rate 25 times as much as changing the entry number from 32 to 64 for 'basicMath'. Such an observation points out the inefficiency of the conventional iTLB design – a majority of iTLB entries is of no avail for most of address-translation requests. However, to avoid the huge mis-recovery penalty, iTLB entries are usually aggressively provisioned, even most of address-translation requests can be satisfied with a small portion of entries at most of the execution time, and further increasing them can only bring in a non-distinctive improvement on the hit rate.

5.2.3 Leakage Efficient iTLB Structure

The over-provisioned iTLB entries, combined with the high locality in instruction streams, lay the foundation of our leakage efficient iTLB design. Here, a leakage efficient iTLB structure is proposed. By introducing the idea of hierarchy design, we insert a small size storage component, which keeps the recent address-translation information, between the processor and the iTLB to filter out unnecessary iTLB accesses. Fig.5.2 compares the conventional iTLB structure and the proposed structure which uses a 1-entry buffer as the higher hierarchy. In the figure, dash lines present paths only being executed when misses in the higher hierarchy happen. To reduce the leakage, the iTLB itself is designed capable of being put into the low-leakage mode when in idle state and restored to the active mode only when necessary. As shown in the figure, the average 98% hit rate of the higher hierarchy (1-entry buffer in the figure) guarantees the time duration in the low-leakage mode, since the iTLB now becomes an extremely inactive component.

Note that, misses in the higher hierarchy lead to accesses to the iTLB instead of iTLB miss exceptions. Comparing with small-sized iTLB configurations, the proposed structure does not incur

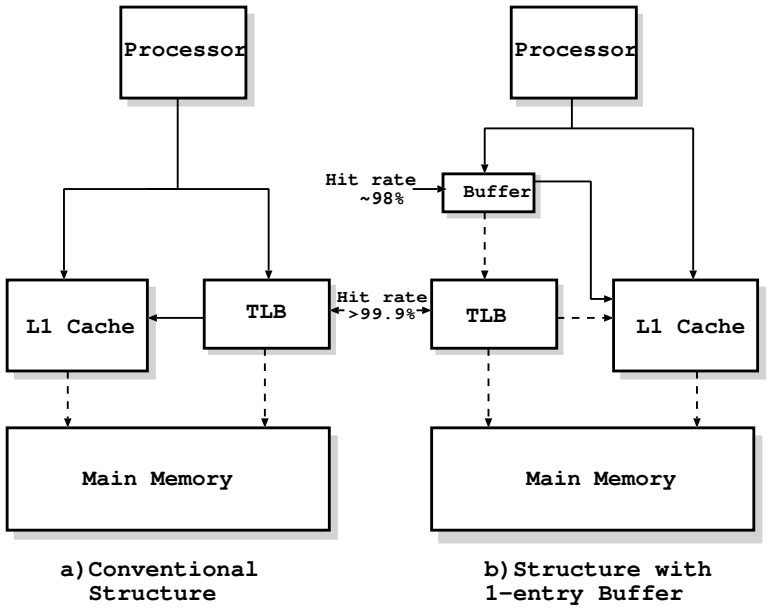


Figure 5.2: Structure of the conventional iTLB and the leakage efficient iTLB

any extra iTLB misses.

In addition, proposed structure is implementation-friendly. Since the leakage control is based on the whole-TLB granularity, proposed structure can be implemented with existing iTLB Intellectual Property (IP), with only minor modifications on the external power rail (see details in subsection 5.3.2). Comparing with the structure using entry/line granularity leakage control [48] [97], proposed one is more suitable for the IP-reuse design methodology (detailed comparisons can be found in Section 5.5).

5.3 Implementation

Circuit-level leakage reduction techniques are usually not a non-overhead one. Transitions from the low-leakage mode to the active mode incur considerable overheads on both performance and power consumption (detailed mode-transition penalties will be shown in the end of subsection 5.3.2). To maximize leakage saving while minimizing the impact to performance, in this section, we discuss how to switch the iTLB between the low-leakage mode the active mode at the appropriate time and in the appropriate manner.

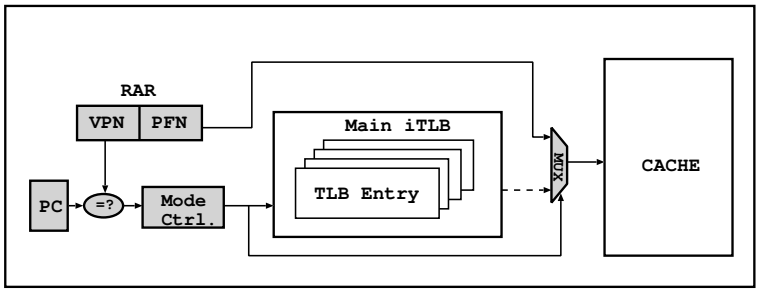
5.3.1 Leakage Control Policies

Based on the proposed iTLB structure, three different leakage control policies are proposed in this subsection.

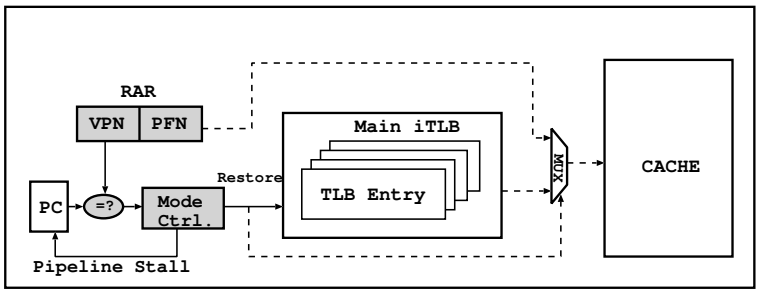
1-RAR Policy: A Recently Accessed Register (RAR), which contains the latest address-translation information, is employed as the higher hierarchy to filter out unnecessary main iTLB accesses[1]. As shown in Fig.3.5, the RAR has the same structure as a TLB entry, holding 64 bits for Virtual Page Number (VPN), Physical Frame Number (PFN), Process ID (PID), flag bits and 14 reserved bits. When the program enters a physical page, the page information is stored in the RAR. As long as the following instructions are fetched from the same page, address translations can be done in the RAR without accessing the main iTLB. If the idle state of the main iTLB has lasted for a certain time (which will be discussed later), we assume such a same-page-hit behavior will continue. Then, the main iTLB can be put into low leakage mode.

Fig.5.3 illustrates an example of how the 1-RAR policy works, where shading blocks indicate being-utilized elements, and the dash lines present the non-exist paths under a given situation. As shown in figures, the working states of the 1-RAR policy can be classified into 3 cases: RAR-hit, RAR-miss while the main iTLB is in the low-leakage mode, and RAR-miss while the main iTLB is in the active mode. The working flow of 1-RAR policy is as follows: (a) the virtual address generated by the processor is compared with the VPN of the RAR. A RAR-hit means the current instruction is in the same page as the preceding one. Therefore, the PFN stored in the RAR will be used as the physical address while skipping the main iTLB look-up. (b) The main iTLB look-up is needed only when the RAR-miss happens. In such a case, if the main iTLB has already been in the low-leakage mode, the processor pipeline will be stalled and the main iTLB must be restored to the active mode first. (c) The main iTLB look-up follows the common routine, except for the requisite of the RAR-update after each main iTLB look-up. Because the RAR is implemented with flip-flops, the RAR comparison can be executed as soon as the PC is updated. When a RAR-miss happens, the main iTLB wake-up can be triggered immediately, and only one clock cycle penalty is incurred for RAR-miss but iTLB-hit references (details will be discussed at the end of this section).

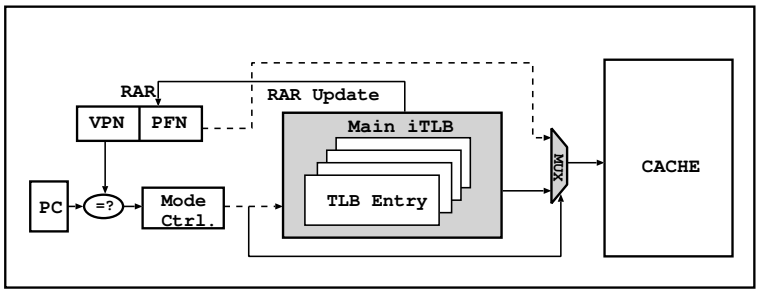
[1]From now on, the lower hierarchy, or the conventional iTLB, in the proposed structure is referred to as the main iTLB



(a) RAR-hit



(b) RAR-miss while the main iTLB is in the low-leakage mode



(c) RAR-miss while the main iTLB is in the active mode

Figure 5.3: Working Process of the 1-RAR Policy

The time between the first RAR-hit and the time-point when the main iTLB goes into low leakage mode is also an important concern, which is referred to as the preliminary time in this thesis. The preliminary time provides a mechanism to avoid mode transitions caused by temporary page-crossing references, which may degrade the leakage reduction result and the performance remarkably. Another consideration of the preliminary time is related to the TLB management scheme of the OS. When process switching occurs, the OS may choose to flush all TLB entries, which usually changes the TLB referencing behavior drastically. Under such a circumstance, selecting a suitable preliminary time, and waiting until the TLB referencing behaviors becoming steady, can eliminate unnecessary overheads caused by the immature mode transitions. In next section, detailed discussions with

experimental results will be presented.

2-RARs Policy: A potential shortcoming of the 1-RAR policy is its incompetence of handling loops that cross the page boundary. Under such circumstances, the contents of RAR will be kept updating, and the main iTLB itself may be waggled between the low-leakage mode and the active mode constantly. Hence, significant overheads on both performance and power may be induced. A simple policy is to use 2 RARs instead of one. The working process of the 2-RARs policy is identical to 1-RAR's except for the RAR updating. In this work, when a RAR updating happens, the latest accessed RAR will be kept while the previous one will be replaced.

Concatenation Policy: The 2-RAR policy incurs extra leakage power because of the second 64-bit register. Here, a Concatenation policy is also proposed to approximate the 2-RAR's performance with only one extra register. To generate the second address-translation information, the VPN and the PFN in the RAR are served as base addresses, and the reserved 14 bits, which are divided into 2 parts: 6 bits for VPN ($Offset_{vpn}$) and 8 bits for PFN ($Offset_{pfn}$), are served as address offsets. When the distance between the current VPN and preceding one is less than 2^6 page size, and the PFN distance is less than 2^8 page size, the second address-translation information can be calculated by following expressions:

$$VPN_{sec} = \{VPN[19:6], Offset_{VPN}\},$$

$$PFN_{sec} = \{PFN[19:8], Offset_{PFN}\}$$

The RAR updating for the concatenation policy is more complex. In this thesis, we classify page-crossing references into two categories: (a) two successive instructions which sit on different pages (we call it WALKING); and (b) branch or jump instructions whose target address locates on another page (referred to as JUMPING). Since changing the base address will also invalidate the address offsets, a conservative base address updating policy is adopted: only WALKING references or iTLB misses can trigger the base address updating, otherwise only the address offsets will be updated. To simplify the hardware complexity, the compiler is employed to detect all WALKING references, and an explicit bit is inserted into instruction set to tell iTLB whether the current reference is a WALKING one.

5.3.2 Hardware Support

Leakage-efficient design needs the support from circuit level. Circuit-level leakage reduction techniques, which are suitable for the proposed design, should satisfy two requests: the state of circuits should be kept when in low-leakage mode; and the mode-transition penalty should be small. In this thesis, the Dual Voltage Supply (DVS) technique is integrated into the main iTLB design to reduce the leakage power of both the iTLB entries and their periphery comparison circuits. While voltage scaling has by wildly used for dynamic power reduction, short channel effects also make it very effective for leakage reduction [48]. When the main iTLB is predicted unnecessary to be accessed in

the near future, it can be put into the lower voltage mode or drowsy mode. By fine tuning the supply voltage in the drowsy mode, data stored in main iTLB entries can be reserved.

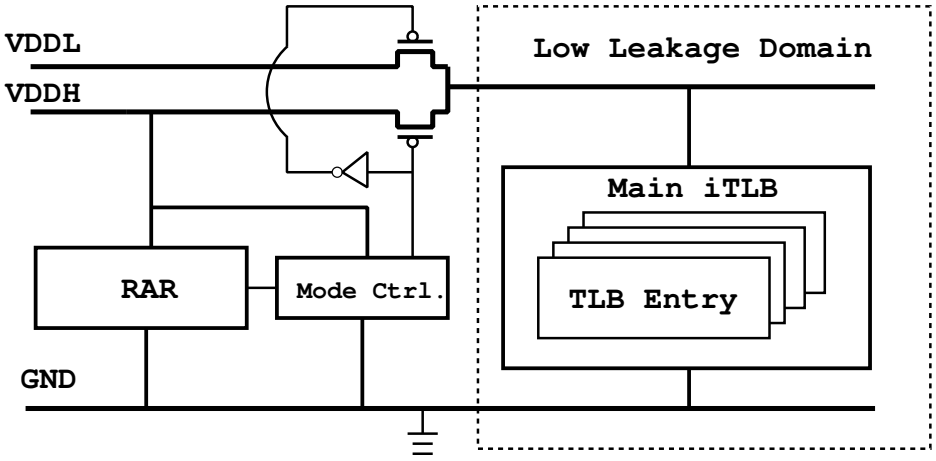


Figure 5.4: iTLB design with the DVS technique

As shown in Fig.5.4, a dual supply network is employed to provide fast switching between different supply voltages. Header PMOS transistors with complementary control signals are used to select between the normal supply voltage (VDDH) and the lower supply voltage (VDDL). Note that, when in drowsy mode, the main iTLB does not allow to be accessed until being restored to the normal voltage. When selecting a $32\lambda^2$ header PMOS, the transition time from the drowsy mode to the active mode for a 16-entry iTLB is about 3.5ns, which is less than one clock cycle for our 200MHz target frequency. In the following sections, the mode transition penalty will be designated as one clock cycle.

The soft error rate [98] is also a concern in very deep sub-micrometer technology. Since the soft error rate increases as supply voltage scaling, to alleviate the side effect caused by lowering voltage, a rather conserved voltage: 0.8v, is selected as the lower voltage supply for the main iTLB.

Table 6.3 presents the power parameters of the proposed iTLB design, which are obtained from the post-layout simulations. We have implemented a 16-entry iTLB using Fujitsu 65nm CMOS technology by Synopsys EDA tools (Design Compiler and Astro) [69]. The implemented iTLB obeys the specification of the MIPS R3000 processor [65], which employs 64 bits entry structure and is designed to cooperate with virtual-index physical-tag caches. Since the post-layout simulation is slow, emulating all iTLB references will be desperately time costing. Here, we intercept a 5000-reference fraction of 'SHA', which shows moderate locality in 8 applications[3], and treat the power consumption of such a fraction as the average power of the whole application programs. In Table 6.3, both dynamic power and leakage power with the normal supply voltage are obtained by PowerCompiler, while the leakage power consumption at the drowsy mode comes from the simulation

[2] λ equals to the half of the minimum transistor channel length

[3] As shown in Fig.5.1, the one-entry miss rate of 'SHA' is in the 4th place of all 8 applications.

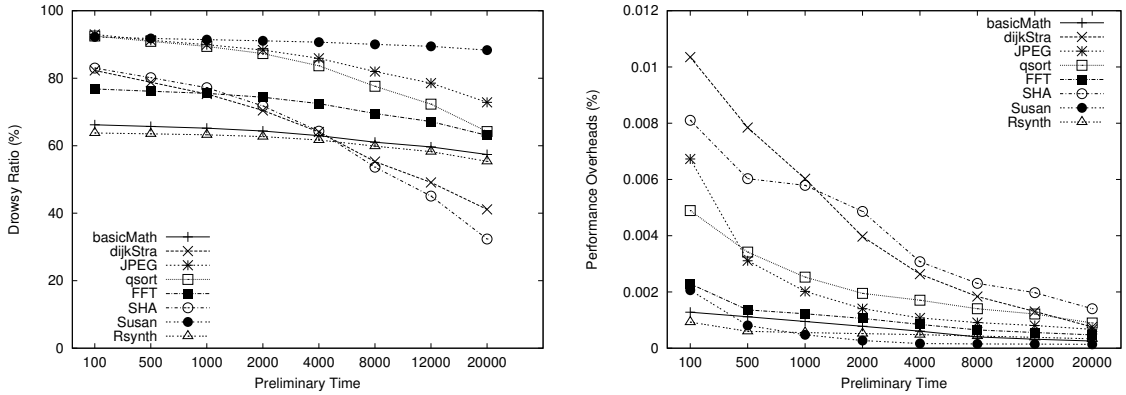
result by HSIM [69].

Table 5.3: Power Parameters(@25°C)

Leakage	
Normal 16-entry	37.8μW
Drowsy 16-entry (0.8v)	9.9μW
Normal 1-entry	3.3μW
Dynamic	
16-entry	688.6μW
RAR	14.1μW
Concatenation	17.4μW

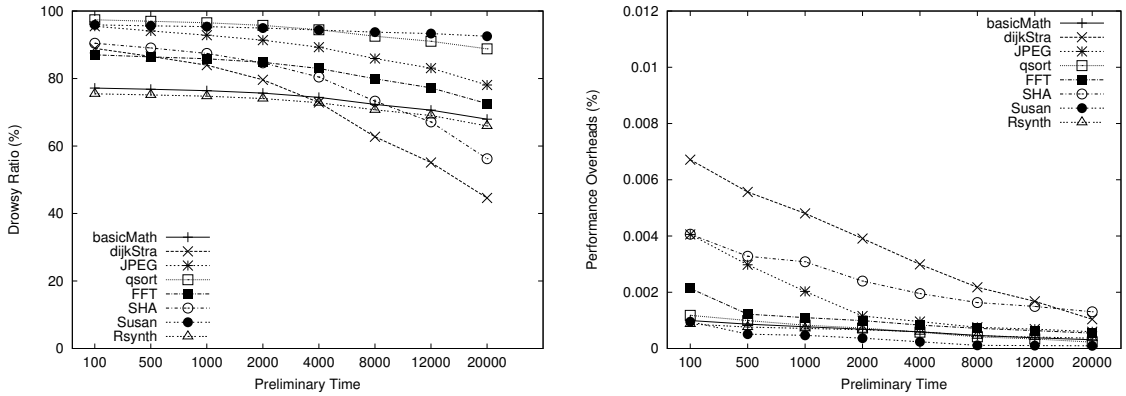
5.4 Evaluation Results

The leakage reduction effects of the proposed design are evaluated in this section. The drowsy ratio, which is the ratio of aggregated drowsy time to the execution time of a program, is combined with performance overheads to measure the efficiency of leakage control polices mentioned in last section. In this chapter, all evaluations is based on the experimental infrastructure presented in Section 5.2, and the base-line configuration is a 16-entry iTLB, which is usually the minimum size for embedded processors. Power evaluation models are also proposed. After selecting the suitable design parameters, the final leakage reduction effects are presented. Note that, the proposed design can be easily implanted to low dynamic power design. In the end of subsection 5.4.1, an approximate dynamic power reduction result is also presented.



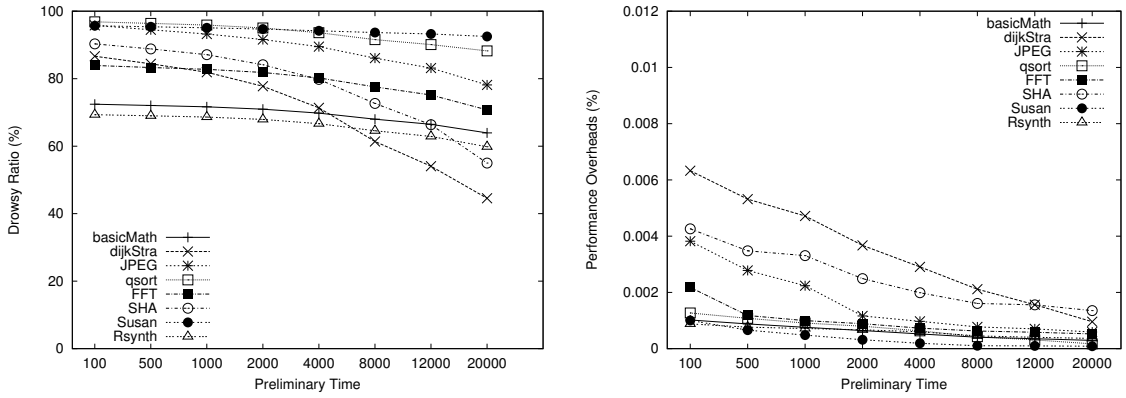
(a) Drowsy Ratio (b) Performance Overheads

Figure 5.5: 1-RAR Policy



(a) Drowsy Ratio (b) Performance Overheads

Figure 5.6: 2-RAR Policy



(a) Drowsy Ratio

(b) Overheads

Figure 5.7: Concatenation Policy

5.4.1 Basic Evaluation

Fig.5.5(a)~Fig.5.7(b) show the drowsy ratio and performance overheads of the 1-RAR, 2-RARs and the Concatenation policy by varying the preliminary time from 100 clock cycles to 20000 clock cycles (assuming the CPI equals to 1). As shown in these figures, the drowsy ratio is kept decreasing as the preliminary time increases, and so does the performance overheads. As mentioned before, an aggressively short preliminary time may be incapable of recognizing the temporary page-crossing references and degrade the performance by triggering short-duration drowsy events; while a conservatively long preliminary time may destroy the drowsy opportunity considerably. It is worth noting that before 8000 clock cycles the decreasing speed of drowsy ratio is not as fast as that of performance overheads. In this chapter, 4000 clock cycles, which can make a good trade-off, is selected for the final power evaluation.

Among three policies, the drowsy ratio of the 2-RARs policy is 1%~5% ahead of the Concatenation policy according to application programs. Meanwhile, the 2-RARs policy also outperforms the 1-RAR policy in terms of drowsy ratio (averagely 8%) because it better fits the footprint of the temporary instruction-fetching and better performs with page-crossing loops.

Power reduction effects are calculated with the drowsy ratio, the number of mode switches, and power parameters presented in Section 5.3. The power evaluation model can be expressed as following:

$$L_{new} = L_{filter} + (1 - P_{Drowsy}) \times L_{iTLB} + P_{Drowsy} \times L_{iTLBL} + L_{counter}, \tag{5.1}$$

$$D_{new} = D_{filter} + (1 - P_{Drowsy}) \times D_{iTLB} + D_{counter} + D_{transition}, \tag{5.2}$$

In equation5.1, L_{new} is the leakage power consumption of a proposed policy; L_{filter} is the leakage power of the higher hierarchy component, which can be one RAR, two RARs, or the Concatenation

register; P_{Drowsy} is the drowsy ratio presented in a percentage form; L_{iTLB} and L_{iTLBL} are leakage power consumption of the main iTLB when in the active mode and the drowsy mode, respectively. Since the preliminary time is selected as 4000 clock-cycle, a 12-bit global counter is also needed, and its leakage power is expressed as $L_{counter}$. In equation5.2, D_{new}, D_{filter}, D_{iTLB} and $D_{counter}$ are the dynamic power counterpart of the equation5.1, while the $D_{transition}$ is the dynamic power consumed by drowsy-to-active mode transitions.

Fig.5.8 and Fig.5.9 show final leakage reduction effects of the iTLB with a 4000 clock-cycle preliminary time. The dynamic power and leakage power of a 4000-clock-cycle-counter are 3.4μW and 0.308μW, respectively. The energy dissipation of each mode transition, which is obtained from post-layout simulation with HSIM is around 2.06×10^{-12}J. $D_{transition}$ can be expressed as following:

$$D_{transition} = \frac{N_{transition}}{\frac{clock\_cycles}{program}} \times \frac{E_{transition}}{\frac{seconds}{clock\_cycle}}, \tag{5.3}$$

Where, the $N_{transition}$ and the $E_{transition}$ indicate the number of mode transitions and the energy dissipation for each mode transition respectively. Note that, since each mode transition incurs one clock cycle penalty, the first part of the equation(3) equals to the performance overheads, which, as shown in Fig.5.5(b) ~ Fig.5.7(b), are less than 0.01%. Therefore, the mode transitions with proposed policies only have a negligible contribution to the total dynamic power.

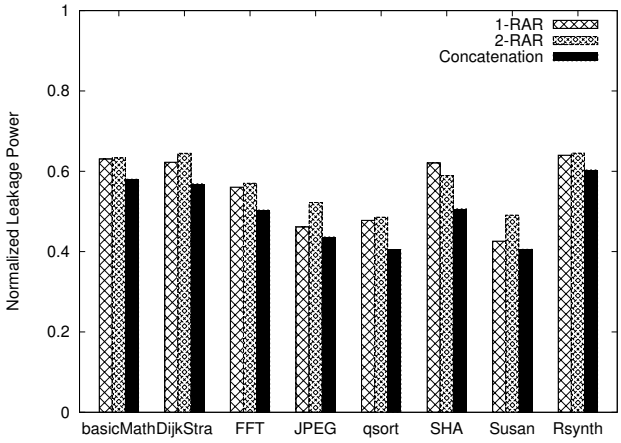


Figure 5.8: Normalized Leakage Power Consumption

As shown in Fig.5.8 and Fig.5.9, while the Concatenation policy has the best leakage reduction effect, dynamic power saving results are highly dependent on the application programs, and the 2-RARs policy slightly outperforms the Concatenation policy. If the spatial locality of an application program is high, for example 'Susan', the 1-RAR policy may have a better performance than 2-RARs in terms of leakage saving because of the extra leakage induced by the second 64bits register of the 2-RARs policy. Averagely, proposed policies can save as much as 50% of the leakage power of iTLB and 75% of the dynamic power, with the performance degradation less than 0.01%.

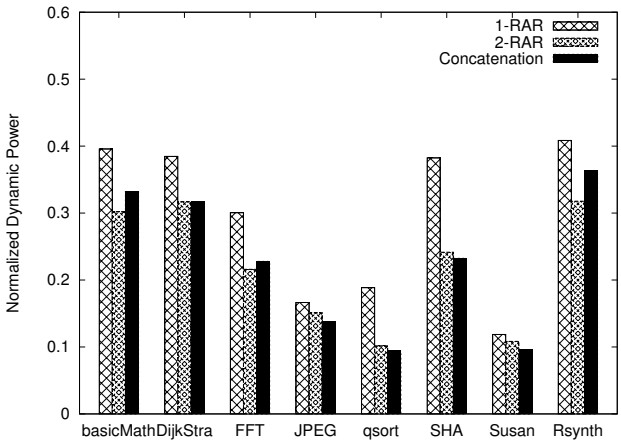


Figure 5.9: Normalized Dynamic Power Consumption

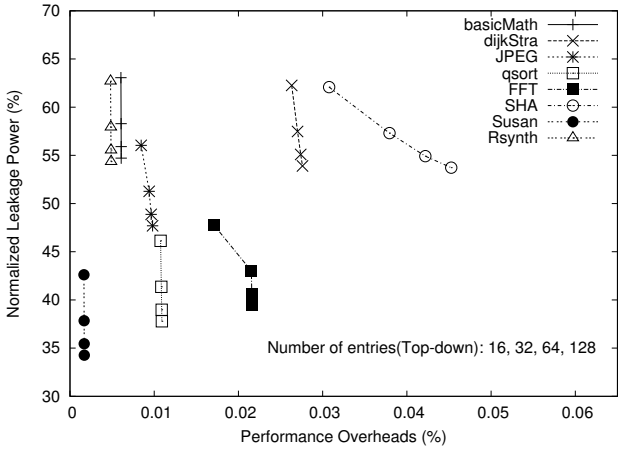
5.4.2 Design Scalability

All above evaluation results are based on a 16-entry base-line configuration. To verify the scalability of the proposed design, additional evaluations are also executed by varying the size of iTLB.

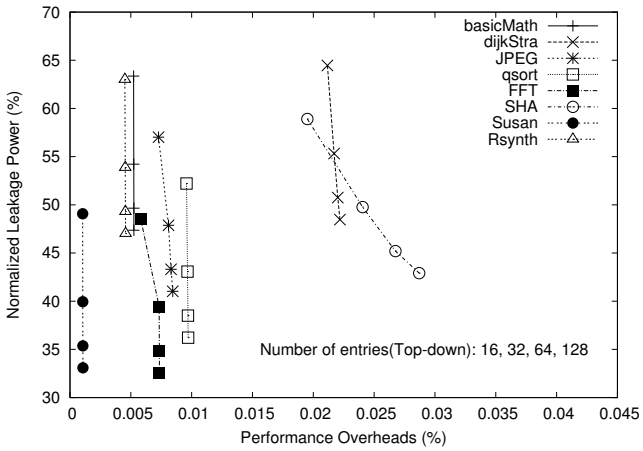
Fig.5.10(a)~Fig.5.10(c) show the robustness testing results with 3 different policies, where the horizontal axis presents the performance overheads and the vertical axis presents the normalized leakage power consumption. Each point on these figures presents a "performance overheads, normalized leakage power" pair of a given application under a specific configuration, which changes from 16-entry to 128-entry in the top-to-bottom order. The normalized leakage power is obtained with the power evaluation model presented in the last subsection, with the L_{iTLB} and L_{iTLBL} scaled by a size-factor, which equals to the current iTLB size divided by 16; and a 4000-cycle preliminary time is selected for all configurations.

A general trend can be observed from these figures – as the TLB size increases, more significant leakage reduction effects can be achieved at a cost of mild performance degradation. This is because the drowsy ratio depends on the referencing patten of application programs rather than the iTLB size. Hence, more leakage power can be saved by putting a larger size main iTLB into the drowsy mode. On the other hand, a large-sized iTLB reduces the number of iTLB misses and shortens the execution time of applications. Taking the testing result of the 1-RAR policy as an example, as the number of entries increasing, the decrease of normalized leakage power mainly comes from the reduced share of RAR's leakage power; while the performance degradation caused by the shortened execution time. Since the performance overheads are highly correlated with the working set of the given applications, if the footprint of an application fits well with the small size iTLB (for instance 'Susan'), a steep line can be observed.

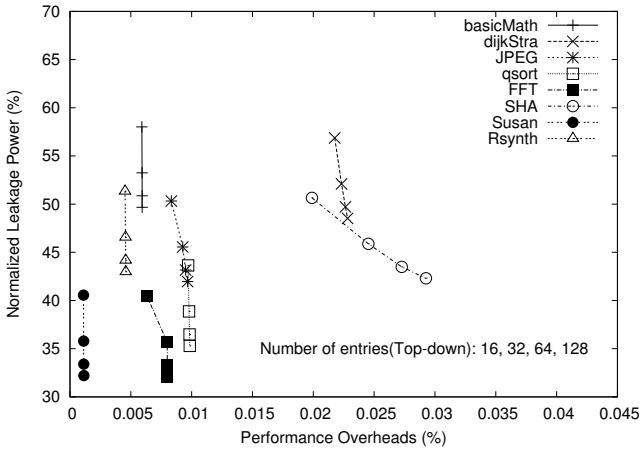
Note that, the Concatenation policy achieves the best leakage reduction effects among 3 polices, especially for the small-sized configuration; while for large-sized configuration, the difference between the 2-RARs solution and the Concatenation solution becomes ambiguous, as the impact of



(a) 1-RAR Policy



(b) 2-RARs Policy



(c) Concatenation Policy

Figure 5.10: Leakage Reduction Efficiency with Varying a iTLB Size

the extra leakage power of the second RAR becomes less significant in the leakage power equation for large-sized iTLBs. As shown in Fig.5.1, increasing iTLB size beyond 16 can only bring in an insignificant iTLB miss rate reduction; thus, a conclusion can be drawn safely that the larger the iTLB is, the better leakage reduction efficiency the proposed design can achieve.

5.5 Related Work

5.5.1 Block Buffering

Previous researches on low-power TLB design are mainly focused on dynamic power. One of the low-power TLB structure is the block buffering [99] [90], where a small number of block-buffers are inserted before the main TLB. If two sequential TLB references are located in the same page, the physical address can be generated directly from the block-buffer, without accessing the main TLB. The power saving of the block-buffering is achieved by employing the clock gating technique, which is fairly transparent from a design and implementation perspective. However, for leakage saving, the mode transitions incur non-negligible penalties on both performance and power consumption due to the imperfect nature of the circuit-level techniques (e.g. DVS and power gating). Therefore, the mode-transition behavior must be managed explicitly and discreetly. As for the proposed design, although it has a similar structure as the block-buffering structure, the mode-control policies and hardware implementation are totally different.

5.5.2 Banking TLB

The banked TLB [91] is another low-power TLB structure. It partitions the main TLB into several banks. By accessing only one bank, this structure can effectively reduce the power consumption per TLB access. The drawback of banked TLB is performance degradation due to the tendency to encounter more capacity misses in specific banks. Paper [100] [101] tried to overcome such a drawback by integration the banked TLB and block-buffers. These schemes can selectively access block buffers or TLB banks according to low bits of the referencing address, and circuit-level techniques have also been proposed to remove the comparison latency from the conventional block buffering. However, the bank-selective mechanism of banked TLB may not be appropriate for leakage reduction in that banks staying in the low-leakage mode can not be restored to the active mode instantaneously. Further, since the high locality of iTLB references, keeping a rather large bank active instead of one or two RAR may be too costly for leakage saving.

5.5.3 Drowsy Cache

A possible alternative scheme for leakage efficient iTLB design is to use the low-leakage mechanism of the drowsy cache [48]. Drowsy cache was proposed to reduce the leakage power of the data cache. By periodically put all cache lines into drowsy mode and active cache lines only when being accessed, the drowsy cache can save the leakage power of a 32KB L1 cache by 52% on average. However, unlike the cache design, the VPN part (as the tag for caches) occupies a significant portion of the whole iTLB. If the VPN part of all entries is activated when a TLB-miss occurs, and such an active state is kept until the next drowsy window, the leakage reduction opportunity will be damaged significantly. Fig.6.12 presents the power reduction effects of a 16-entry iTLB which simply adopts

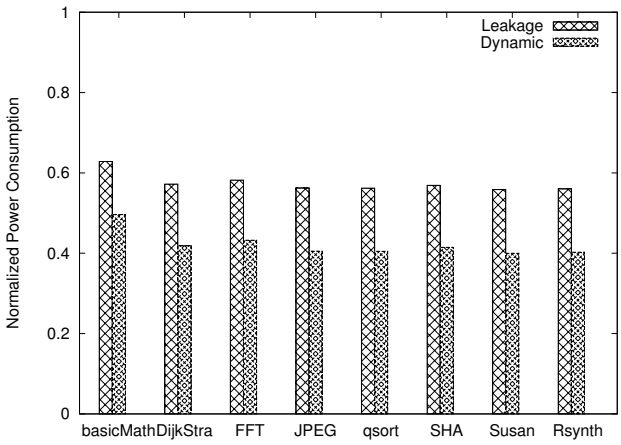


Figure 5.11: Normalized Power Consumption with Drowsy Cache Structure: Leakage & Dynamic

the leakage reduction mechanism of drowsy cache. Here, the drowsy window is 4000 clock cycles, and we assume the activation of the VPN part consumes no extra power. As shown in the Fig.6.12, leakage power of the iTLB is reduced by 43% on average, and dynamic power is saved by 57%. Comparing with Fig.5.8 and Fig.5.9, proposed leakage efficient iTLB design outperforms the one with drowsy cache mechanism by 7% in terms of leakage power reduction and 18% for dynamic power reduction. Furthermore, by adopting the mechanism of drowsy cache, the leakage control must be implemented in each-entry granularity, which will complicate layout design significantly, and the exiting IP of iTLB must be re-designed. Paper [97] proposed an improved drowsy cache design, which employs a small L0 cache to reduce the access-frequency to the L1 drowsy cache. Although it seems to have a similar structure as the one proposed in this chapter, its leakage saving effects come from the each-entry leakage control mechanism of the drowsy cache, instead of the whole iTLB leakage control policies as in ours.

5.5.4 Dual-Vth Design

Employing dual-Vth cells, i.e. low-Vth cells for the RAR while high-Vth cells for the main iTLB, is also an option to reduce the leakage power of the iTLB (such a design is referred as the Dual-Vth design in this thesis). The Dual-Vth design is implementation-friendly, and its leakage reduction effect can be significant. However, the performance degradation incurred by such a design may be much higher than ours. Here, a comparison is made the Dual-Vth design and ours. Both of the design follow the based-line configuration as shown in Section 5.4, while the main iTLB of the Dual-Vth design is implemented with Fujitsu's high-Vth Lib.(202MNC). Evaluation results show that the Dual-Vth design can reduce the leakage power by 60% at a cost of 16% or 2% performance degradation according to two different main iTLB design strategies.

1) The main iTLB is designed to be accessed in one clock cycle.

Since the RAR is accessed right after the virtual address generation, in this case, the pipeline does

not need to be stalled. However, the slower but less leaky main iTLB may degrade the maximum clock frequency of the processor in that iTLB usually sits in the critical path. The evaluation results show that the Dual-Vth design may incur a 16% speed-down.

2) The main iTLB can have a multiple-cycle access.

In this case, the frequency degradation can be avoided, while the performance degradation will be introduced by the stalled pipeline. For example, if the main iTLB access takes two clock cycles, we need to stall the pipeline for one cycle for each RAR miss. Now, the performance degradation equals to the RAR miss rate, which is 2% in average as shown in Fig.5.1. On the other hand, our design will not degrade the maximum clock frequency, and the pipeline will be stalled only when a RAR miss happens and the main iTLB is in the drowsy mode at the same time.

Although the Dual-Vth design can reduce more leakage power, the performance degradation caused may offset such an advantage from the perspective of energy dissipation. As illustrated above, the performance degradation of the Dual-Vth design comes from the stalled pipeline, and it will affect the execution time of the whole processor directly. Thus, additional leakage energy will be dissipated not only by the iTLB itself but also by the other parts of the processor. Fairly, the additional leakage energy should be treated as the energy overheads of the Dual-Vth design, and its normalized leakage energy, E_{L_norm}, can be expressed as:

$$E_{L\_norm} = \frac{E_{L\_new}}{E_{L\_iTLB}} = \frac{L_{new} \times T_{orig} + L_{processor} \times T_{addi}}{L_{iTLB} \times T_{orig}} = \frac{L_{new}}{L_{iTLB}} + \frac{L_{processor}}{L_{iTLB}} \times \frac{T_{addi}}{T_{orig}}, \tag{5.4}$$

Where, E_{L_new} is the leakage energy of the Dual-Vth design, E_{L_iTLB} is the leakage energy of a common 16-entry iTLB. L_{new}, L_{iTLB}, and $L_{processor}$ are the leakage power of the Dual-Vth design, the common iTLB, and the whole processor, respectively; while T_{orig} and T_{addi} are the execution time of the processor with the common iTLB and the additional execution time caused by the Dual-Vth design.

According to the equation, the normalized leakage energy of the Dual-Vth design can be expressed as a function of the leakage power reduction achieved, performance degradation caused, and the leakage power share of the iTLB to the whole processor.

When being applied to an aforementioned MIPS R3000 processor (5-stage pipeline, 8K 2-way instruction and data caches), where the leakage power of the iTLB is 11% of the whole processor, the Dual-Vth design can reduce the leakage energy of the iTLB by 42%.

The above equation can also be applied safely to calculate the normalized leakage energy of our design. Since our design incurs 0.01% performance degradation, there is only a negligible difference between its normalized leakage power and normalized leakage energy, which is still 50%. Note that, since the performance degradation incurred by our design is only 0.01%, the difference between its normalized leakage power and normalized leakage energy is negligible. As a result, our design reduces 8% more leakage energy than the Dual-Vth design and incurs less performance degradation.

5.6 Conclusions

A leakage efficient iTLB design has been proposed. By exploiting the iTLB referencing locality, a small higher hierarchy component is inserted between the processor and the main iTLB to filter out unnecessary main iTLB accesses. With the integration of the dual voltage supply technique, the main iTLB can be turned into the low-leakage mode for leakage saving when predicted not to be accessed in the near future. The proposed design also can be utilized to reduce dynamic power with the help of clock gating technique. Evaluation results show, 50% of the leakage power and 75% of the dynamic power can be reduced, at a cost of 0.01% performance degradation.

The working efficiency of proposed iTLB design is highly dependent on the locality existing in instruction stream. The evaluation results show in Section 5.4 and Section 5.5 are based on typical embedded applications. As mentioned before, the page transition for instructions is mostly caused by function calls and returns, so for other applications, for example, the program written by function programming language (LISP [102] as an example), the locality of instruction may be changed and proposed design may be inappropriate, but the leakage-efficient iTLB design for those applications is out of the range of this study.

Chapter 6

Leakage-efficient Data TLB

6.1 Introduction

In this chapter, a leakage-efficient data TLB design is proposed. The data TLB (dTLB) has a similar structure as instruction TLB's, and for the same reason as mentioned in Section 5.1, the dTLB consumes a significant leakage share in an embedded processor – as shown Chapter 3, a 16-entry dTLB consume as much as 14% of the leakage power of a MIPS R3000 processor. Although TLBs have a cache-like structure, blindly transplanting cache leakage reduction mechanisms, such as cache decay [47] and drowsy cache [48], into a dTLB design will introduce unacceptable performance and power overheads, due to their different access pattern, replacement policy and mis-recover penalty. Further, the dTLB is one of the most active components in embedded processors with a high utilization, which intuitively does not leave much space for leakage reduction. Fortunately, page-based dTLB references exhibit a high degree of locality, implying that, in a short time period, only a small subset of dTLB entries actually serves for the data-address-translation requests. Here, we observe the dTLB referencing in a finer time resolution. By dividing the overall execution time into smaller time slices, the locality of dTLB references in and between adjacent slices can be utilized and contributive dTLB entries[1] can be recognized. Then, with integration of the Dual Voltage Supply (DVS) technique, those non-contributive entries can be put into low leakage mode (with the lower voltage supply) dynamically. Based on such a design philosophy, a leakage efficient dTLB design is proposed. Power evaluation results show that the proposed design can reduce the leakage power of a dTLB by 37% with negligible performance degradation.

It is worth noting that although we focus on reducing leakage power of the dTLB, the proposed design can be easily integrated with the clock gating technique to reduce the dynamic power as well. According to the power evaluation result, 65% of the dynamic power of the dTLB can also be reduced. To the best of our knowledge, no previous work can provide such an uniform low power dTLB solution.

[1] In this chapter, contributive entries mean dTLB entries that actually serve for the address translation within each time slice, and other entries are referred to as non-contributive entries.

The remainder of this chapter is organized as follows. The next section presents the experimental platform for this work. In Section 6.3, the basic design philosophy and detailed leakage reduction mechanism will be illustrated. Leakage saving results will be shown in Section 6.4, and we will discuss related work in Section 6.5. Section 6.6 concludes the chapter.

6.2 Experimental Setup

The experimental environment used in this chapter is same as the one described in Section 5.2. QEMU [93] is selected as the experimental platform on which trace-driven simulations are executed to capture necessary data. Since TLBs have a tighter connection with the Operating System (OS) than other components in a processor, the OS (Debian) and application programs are running simultaneously on such a platform. A group of authors [94] modified the basic structure of QEMU, so that it can be used to trace TLB references and other necessary information. The configuration parameters are shown in Table 6.1. To better emulate the real working state of a TLB, not only TLB references but also the TLB-flush information is traced by executing eight application programs from different fields of MiBench [71]. The number of TLB-flushes of each application program is shown in Table 6.2. When application programs are executed, several OS related processes are running on the background, and there are also some basic processes, like Shell, are running with application programs concurrently.

Table 6.1: Configuration Parameters

Trace Environment	
CPU Type	MIPS R3000
Instruction Execution	In-order
OS Type	Debian
Kernel	Linux 2.6.15
Shell	ash
Compiler	GCC(4.2.2)

Table 6.2: TLB-flushes of Application Programs

Programs	TLB-flushes	Programs	TLB-flushes
BasicMath	72	Dijkstra	59
JPEG	73	Qsort	32
FFT	49	SHA	87
Susan	52	Rsynth	102

The power evaluation in this chapter is based on the post-layout simulation. We implemented a 16-entry dTLB with Fujitsu 65nm CMOS technology using Synopsys EDA tools (Design Compiler and Astro) [69]. The implemented dTLB obeys the specification of the MIPS R3000 processor [65], which employs 64 bits entry structure and is designed to cooperate with virtual-index physical-tag caches. Since the post-layout simulation is slow, emulating all dTLB references will be desperately time costing. Here, we intercept a 5000-reference fraction of 'SHA', which shows moderate locality in 8 applications, and treat the power consumption of such a fraction as the average power of the

whole application programs. Then, obtained results are used to calculate the final power reduction effects, combined with the utilization ratio of each logic ingredient. The value of both dynamic power and leakage power used in this chapter is obtained with Synopsys' PowerCompiler, while the leakage reduction ratio achieved by DVS and mode transition overheads come from the circuit simulation with HSIM [69].

6.3 Design Philosophy

Leakage efficient design bases on the assumption that a certain fraction of the design target can be put into low leakage mode and restored to the active mode without significantly degrading the performance. The leakage reduction effects rely on the scale of the leakage reduction target, the time duration in low leakage mode and the mode transition frequency. As for the dTLB, which is one of the most active components in embedded processors, there seems to be no enough space left for leakage optimization. However, as the leakage power consumption of the dTLB continues getting prominent, its reduction mechanism becomes indispensable. In this section, we will analyze the dTLB referencing pattern and the corresponding leakage reduction opportunities. Then, based on the analysis results, a leakage efficient dTLB design will be presented.

6.3.1 dTLB Referencing Pattern Analysis

Another metric, the Sequential Page Access Rate (SPAR) which is the ratio of sequential TLB references that hit on the same page divided by the total TLB references, is also introduced to analyze the different references pattern of iTLB and dTLB. Note that, SPAR is equal to 1 minus the 1-entry TLB miss ratio and is an direct indicator of spatial locality. As shown in Fig.5.1 and Fig.6.1, while iTLB references of all applications exhibit a significant spatial locality (the SPAR of all 8 applications is above 95%), the SPAR of dTLB is fluctuating drastically from program to program, and is generally much worse than iTLB's (the SPAR of Susan, which is the worst one in all 8 applications, is less than 20%). Thus, different leakage control methods are requested to adapt for the different referencing pattern of instruction and data.

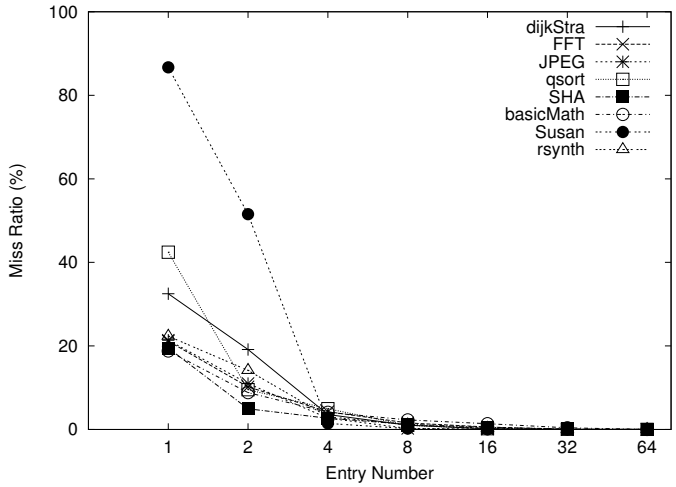


Figure 6.1: dTLB Miss Ratio

Typically, data references exhibit a high degree of temporal locality, indicating that in a short interval of execution, certain memory locations tend to be accessed repeatedly. In general, not all

such locations are spatially close. But from the perspective of the page-based TLB referencing, the number of hit entries in a given interval, which is termed as temporary footprint in this paper, has a high probability to be confined to a small range. Fig.6.2 shows the distribution of the temporary footprint of eight application programs at 4000 clock cycle intervals, where each bar presents the proportion of the intervals whose temporary footprint can be 1~2/3~4/5~6/7~8/more than 8 entries. Although the distribution varies drastically from program to program, a consistent dTLB referencing pattern can be observed, that is, the temporary footprint of the most of intervals only covers a small number of dTLB entries. For instance, the average percentage of intervals whose temporary footprint is bigger than 4 entries is about 13%, and that bigger than 8 entries is less than 1%.

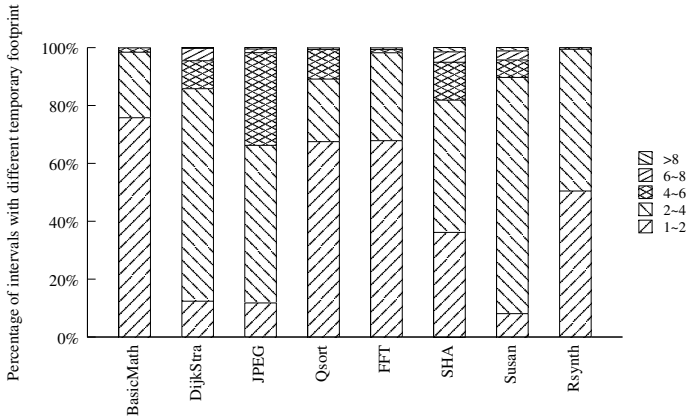


Figure 6.2: Temporary Footprint Distribution

Although only a small subset of dTLB entries actually serves for the virtual-to-physical address translation in each execution interval, to avoid the huge mis-recover penalty, dTLB entries are usually aggressively provisioned in modern embedded processors. The over-provisioned dTLB entries, combined with the interval-based referencing pattern, lay the foundation of our leakage efficient dTLB design – if the temporary footprint can be detected at run-time, by turning those non-contributive entries into the low leakage mode and restoring them back only when necessary, significant leakage reduction effects can be achieved without much performance degradation. In this paper, a 16-entry dTLB, which is usually the minimum size for embedded processors, is selected as the basic configuration for the purpose of evaluation.

6.3.2 dTLB Leakage Reduction Mechanism

Based on the analysis of the dTLB referencing pattern, we divide the overall execution time of a program into smaller time slices. The dTLB leakage reduction mechanism proposed in this subsection tries to fit the temporary footprint of dTLB references by dynamically changing the number of active dTLB entries in each time slice. After illustrating the basic design philosophy, three design factors

will also be discussed in order to achieve the best leakage reduction efficiency.

One straight forward mechanism to detect the temporary footprint is to put all dTLB entries into the low leakage mode periodically, and a dTLB entry is activated[2] only when being accessed again. Such a mechanism is similar to that used in drowsy cache [48], and in this paper it is referred to as the SD (Simple Drowsy) mechanism. With the SD mechanism, a global time counter is needed and the mode control circuits must be implemented at each entry granularity. Fig.3.5 shows the structure of a dTLB entry, which holds 64 bits for Virtual Page Number (VPN), Physical Frame Number (PFN), Process ID (PID), flag bits and 14 reserved bits. When a virtual address arrives at the dTLB, the SD mechanism works in a 3-step fashion: 1) the highest 20 bits of the virtual data address are compared with the VPN of all active TLB entries. If an entry matches (such a case is referred to as an active-hit and the corresponding miss as an active-miss in the remainder of this paper), its PFN will be concatenated with lower bits of the virtual address to form the physical address, after confirming no exception caused by the PID and flag bits. Therefore, a 16-bit trace register is needed to track the mode state of each entry, and an entry is allowed to be accessed only when the corresponding bit in the trace register is set. 2) In case of active-misses, the processor pipeline will be stalled. At the same time, TLB-TAGs, which store the VPN part of each entry, will be activated. Then, the virtual address will be compared with the VPN of all entries that had not been accessed yet. 3) If a TLB-TAG hit occurs, the PFN will be read out after the hit-entry being fully activated, which incurs a 4 clock cycles penalty (as will be discussed in detail below, the activation process takes one clock cycle); otherwise, a dTLB miss happens, and the whole dTLB will be restored to the active mode. Since the activation process can be overlapped with the TLB miss handle process, only 2 clock cycle penalty is incurred.

As shown in Fig.3.5, the TLB-TAGs account for almost one thirds of the dTLB's size. Since the leakage power is proportional to the number of transistors engaged in a design, if TLB-TAGs of all entries are activated when an active-miss happens, and such an active state is kept until the next slice, the leakage reduction opportunity will be damaged significantly (The power reduction effect of SD mechanism will be presented in Section 6.5). Here, the TLB-TAG and its periphery comparison circuits are designed capable of being accessed in the low leakage mode, without having to be restored to the active mode first. Working with lower voltage will increase the transistor's transition time, therefore decrease the operating speed. However, an active-miss will stall the pipeline, and the TLB-TAG access follows a different path from common TLB accesses. Fig.6.3 presents the basic structure of a dTLB. The solid lines present shared paths for both TLB-TAG accesses and common TLB accesses, while the dash lines are paths only being used by common TLB accesses. As shown in the figure, a common TLB access compares the virtual address with the TLB-TAG first. After checking the PID and flag bits of the hit-entry, the physical address will be formed by concatenation the PFN and lower bits of the virtual address. Since most of modern embedded processors integrated caches with the virtual-index physical-tag style, the comparison of cache tag and PFN, and the cache data selecting are also executed in the same cycle. On the other hand, the TLB-TAG access path

[2]indicating voltage supply is restored to the higher voltage

finishes after the VPN comparison. As such, the path of a TLB-TAG access is much shorter than that of a common TLB access; and by choosing a proper supply voltage, the lower voltage design will not bring in any frequency degradation (detailed discussion will be presented in next subsection). Further, the penalty of an active-miss can be reduced to 3 clock cycles.

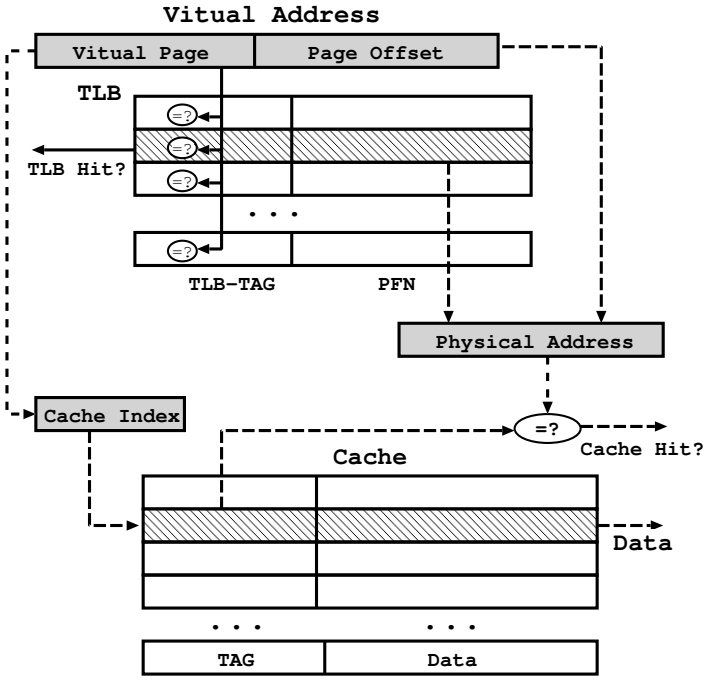


Figure 6.3: TLB-TAG Path

With the above design philosophy, we next discuss three design factors that may influence the final leakage reduction effects.

Fast-wake-up: The spatial locality may have a drastic variation between different program segments, and few time slices may have a rather large temporary footprint. If the spatial locality of a time slice is low, considerable performance and power overheads will be incurred by activating a large number of TLB entries. Here, a fast-wake-up policy is proposed to set the upper limit of performance degradation. If the number of active-misses reaches a preset threshold (referred to as fast-wake-up threshold) in a given time slice, the program segment executed in this slice will be recognized as a low locality segment and the whole dTLB will be activated immediately to eliminated the potential penalties caused by staying in the low leakage mode.

Correlation between Time Slices: Another design concern is the correlation between sequential time slices. If the data references in current slice are highly correlated with those of previous slices; then, keeping state of previous active TLB entries in a new time slice will be helpful in terms of eliminating entry-activation overheads.

Time Slice Length: The time slice length determines how often dTLB entries are put into low leakage mode. A short time slice length induces high-frequency mode transitions. Hence, the

mode transition penalty. While a long time slice length may be unable to reflect the changing of data referencing pattern, and increase the probability of keeping profitless entries active or even a full active dTLB. The detailed discussion of the impact of fast-wake-up threshold, correlation between time slice and time slice length on performance and leakage reduction effects will be presented in next section.

6.3.3 Hardware Support

Leakage-efficient design needs the support from circuit level. Circuit-level leakage reduction techniques, which are suitable for the proposed mechanism, should satisfy two requests: the state of circuits should be kept when in low leakage mode; and the mode transition penalty should be small. In this paper, the Dual Voltage Supply (DVS) technique is integrated into the dTLB design to reduce the leakage power of both the dTLB entries and their periphery comparison circuits. While the voltage scaling has by wildly used for dynamic power reduction, short channel effects also make it very effective for leakage reduction [48]. When dTLB entries are predicted unnecessary to be accessed in the near future, they can be switched to the lower voltage mode or the drowsy mode. By fine tuning the supply voltage in the drowsy mode, data stored in dTLB entries can be reserved.

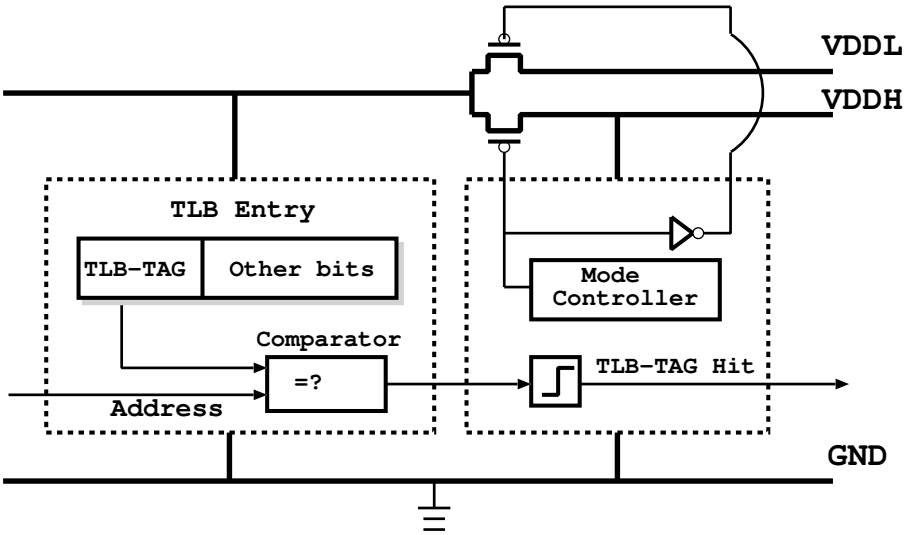


Figure 6.4: dTLB Entry Schematic

As shown in Fig.6.4, a dual supply network is employed to provide fast switching between supply voltages for each dTLB entry. Header PMOS transistors with complementary control signals are used to select between the normal supply voltages (VDDH) and the lower supply voltage (VDDL). Note that, all components working in the drowsy mode, except TLB-TAGs, do not allow to be accessed until being restored to the normal voltage. When selecting $16\lambda^3$ header PMOS with each-entry control granularity, the mode transition time for a dTLB entry is about 1.9ns, which is less than

[3] λ equals to the half of the minimum transistor channel length

one clock cycle for our 200MHz target frequency. In the following subsections, the mode transition penalty will be designated as one clock cycle.

Fig.6.4 shows the schematic of a dTLB entry. Since TLB-TAGs can be accessed in the drowsy mode, a level shifter is appended after the comparison circuit. Voltage scaling will degrade the operating speed, so the VDDL must be carefully selected. Equation6.1 shows the relationship between operating frequency and supply voltage, where V_{norm} and F_{norm} are operating voltage and frequency which are normalized to the maximum voltage V_{max} and frequency F_{max}.

$$V_{norm} = V_{th}/V_{max} + (1 - V_{th}/V_{max}) \times F_{norm}, \tag{6.1}$$

In this paper, we choose the 0.9v as the lower voltage of dTLB entries (while the higher voltage is 1.2v), which means a 40% speed-down. As was mentioned in last subsection, the slower operating does not affect the overall frequency because of the shorter path of the TLB-TAG access. Simulation results have confirmed such an assumption. Table 6.3 lists the power parameters of the proposed design, which are obtained from the post-layout simulation as mentioned in Section 6.2.

Table 6.3: Power Parameters

Leakage	
Active 16-entry	37.8μW
Active 1-entry	3.3μW
Drowsy 1-entry (0.9v)	1.4μW
Dynamic	
16-entry	688.6μW
1-entry	14.1μW

6.4 Evaluation Results

The leakage reduction effects of the proposed design are evaluated in this section. The drowsy ratio, which is the percentage represent of the aggregated drowsy time divided by the execution time of a program, has a direct impact on final power saving results. In this section, the drowsy ratio is combined with performance overheads to measure the leakage reduction efficiency of the proposed design. Power evaluation models are also proposed. After fine-tuning the design parameters, the final leakage reduction effects are presented. Note that, the proposed mechanism can be easily implanted to low dynamic power design. In the end of subsection 6.4.1, an approximate dynamic power reduction result is also presented.

6.4.1 Basic Evaluation

In Section 6.3, we discussed three factors that may affect the dTLB leakage saving results: fast-wake-up, correlation between time slices, and the slice length. In this subsection, evaluation results on how these factors work are presented. Here, the correlation between time slices is measured by the history length, which indicates the active state of how many preceding slices should be kept.

Fig.6.5 and Fig.6.6 show the drowsy ratio and performance overheads of a 16-entry dTLB by varying slice length without considering fast-wake-up or correlation between slices. Although the drowsy ratio and performance overheads change drastically from program to program, a similar trend can be observed from all eight applications: both the drowsy ratio and the performance overheads constantly decrease with the increased slice length. After 4000 clock cycles, the descending speed of performance overheads achieved by expanding slice length can not catch up with the speed of drowsy ratio degradation. In this paper, we choose the 4000 clock cycles as the slice length for the final power evaluation.

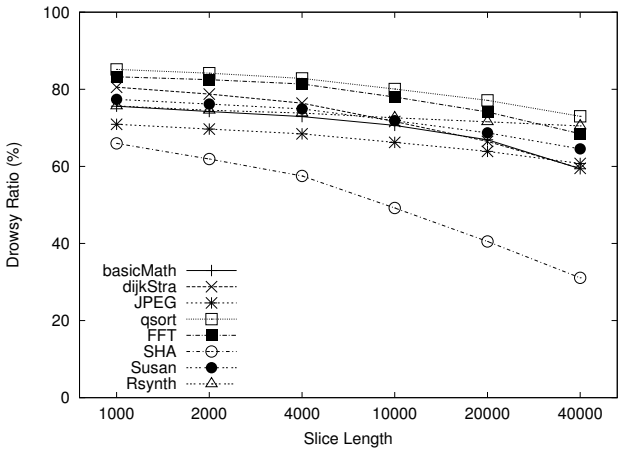


Figure 6.5: dTLB Drowsy Ratio: Slice Length

Fig.6.7 and Fig.6.8 show the drowsy ratio and performance overheads under different (history

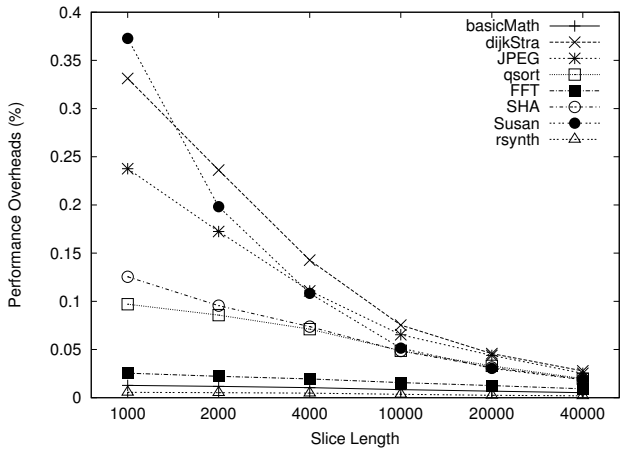


Figure 6.6: dTLB Performance Overheads: Slice Length

length (HL), fast-wake-up threshold (FT)) configurations. In Fig.6.8, the performance overheads continue decreasing as the history length increases, and a similar trend exists between the drowsy ratio and the history length in Fig.6.7, except the (0,2) configuration. A significant improvement on the leakage reduction efficiency can be achieved by changing the history length from 0 to 2 (averagely, the overheads are reduced by 70% with drowsy ratio degradation less than 6%), while further increasing the history length can not sustain such a trend. The fast-wake-up threshold, which sets the upper limit of overheads, has a similar affect on the drowsy ratio and performance overheads but in a less obvious way. In case of the configuration of small fast-wake-up threshold with small history length, the drowsy ratio may be degraded significantly (For example, the drowsy ratio of "dijkstra" at (0,2) is only about 17%). Here, the (2,4) pair, which ensures the worst performance overheads is less 0.3%, is selected for the purpose of power evaluation.

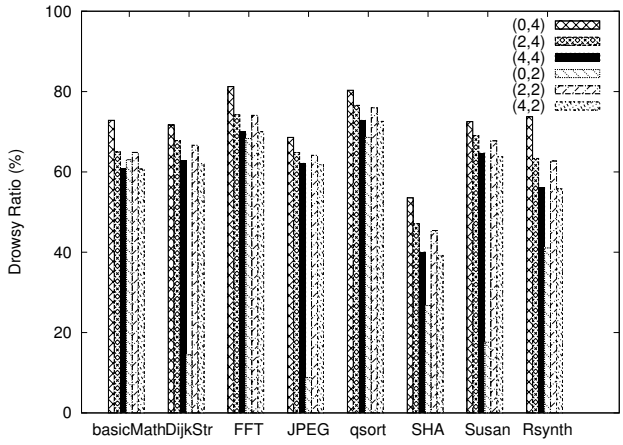


Figure 6.7: dTLB Drowsy Ratio: HL & FT

The power evaluation of the proposed design is complex because of the 3-step fashion design philosophy. To simplify the question, the dynamic power consumed by drowsy TLB-TAG accesses

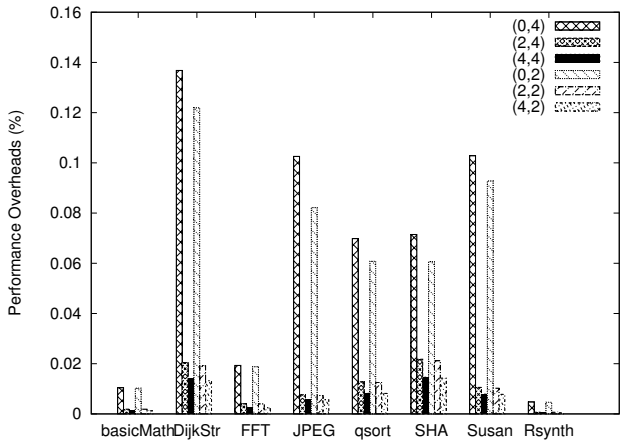


Figure 6.8: dTLB Performance Overheads: HL & FT

is treated the same as full active TLB accesses. The leakage and the dynamic power consumption can be calculated as follows:

$$L_{new} = (1 - P_{Drowsy}) \times L_{entry} \times N_{entry} + L_{counter} + L_{trace} + P_{Drowsy} \times L_{entry_L} \times N_{entry}, \tag{6.2}$$

$$D_{new} = (1 - P_{Drowsy} + P_{Drowsy\_acc}) \times D_{entry} \times N_{entry} + D_{counter} + D_{trace} + D_{transition}, \tag{6.3}$$

In equation6.2, the L_{entry} and L_{entry_L} denote the leakage power consumption of a single dTLB entry working in the active mode and the drowsy mode respectively, and the $L_{counter}$ and L_{trace} are the leakage power consumed by the time counter and the trace register. The N_{entry} is the number of dTLB entries engaged in a design, and the P_{Drowsy} is the drowsy ratio which is shown in Fig.6.7. As mentioned earlier, the L_{entry_L} is measured by 0.9v voltage supply.

In equation6.3, the D_{entry}, $D_{counter}$ and D_{trace} are the dynamic counterpart of the L_{entry}, $L_{counter}$ and L_{trace}; and the P_{Drowsy_acc} is the percentage of drowsy accesses with respect to the number of dTLB references, which equals to one thirds of the performance overheads shown in Fig.6.8.

The $D_{transition}$ is the dynamic power consumed during mode transitions, which can be expressed as the following equation:

$$D_{transition} = \frac{N_{transition}}{T_{exec}} \times E_{transition}, \tag{6.4}$$

Where, the $E_{transition}$ indicates the energy dissipation of one dTLB entry for each mode transition, which is obtained from post-layout simulation with HSIM, and equals to 6.23×10^{-14}J. The $N_{transition}$ is the number of mode transitions which can be caused by a single entry wake-up, a fast-wake-up and a miss handling process when the dTLB is in the drowsy mode. The T_{exec} is execution time of a program. Table 6.4 shows transition ratio which equals to the first part of Equation(6.4) in the percentage form.

Fig.6.9 shows the normalized power consumption of the proposed design on both leakage and dynamic with the 4000 clock-cycle slice length. The dynamic power and leakage power of a 4000-

Table 6.4: Transition Ratio of Application Programs

Programs	Transition Ratio	Programs	Transition Ratio
BasicMath	0.0073%	Dijkstra	0.0328%
JPEG	0.0106%	Qsort	0.0187%
FFT	0.0168%	SHA	0.0830%
Susan	0.0130%	Rsynth	0.0011%

clock-cycle-counter is 3.4μW and 0.308μW, and the trace register consume 3.6μW dynamic power and 0.4μW leakage power respectively. As the evaluation result shows, more than 37% leakage power and 65% dynamic power can be saved by proposed mechanism with performance degradation less than 0.01%.

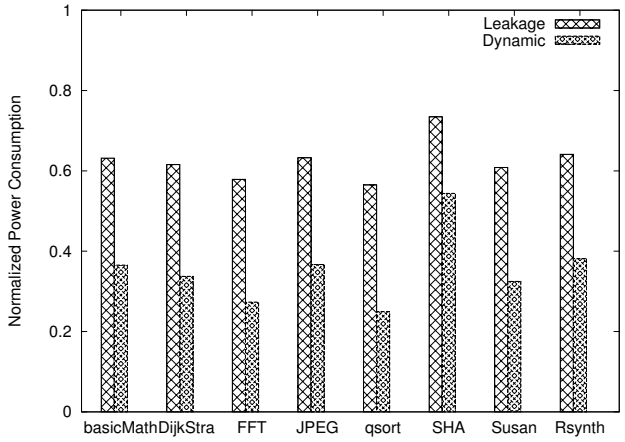


Figure 6.9: Normalized Power Consumption: Leakage & Dynamic

6.4.2 Design Scalability

All above evaluation results are based on a 16-entry dTLB with typical embedded application programs. To verify the scalability of the proposed design, additional evaluations are also executed by varying the size of dTLBs and on data intensive applications.

Fig.6.10 shows the leakage reduction efficiency of the proposed mechanism with different size configurations, where the horizontal axis presents the performance overheads and the vertical axis presents the normalized leakage power consumption. Each point on the figure presents a "performance overheads, normalized leakage power" pair for a given configuration, which changes from 16-entry to 128-entry in the top-to-bottom order. The normalized leakage power is obtained with the power evaluation model presented in the last subsection, and the value of design factors is the same as that of the 16-entry dTLB.

A general trend can be observed from the Fig.6.10– as the dTLB size increases, more significant

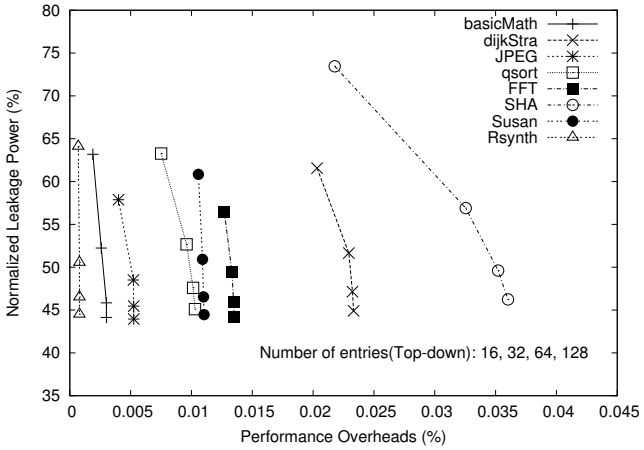


Figure 6.10: Leakage Reduction Efficiency with Varying size dTLBs

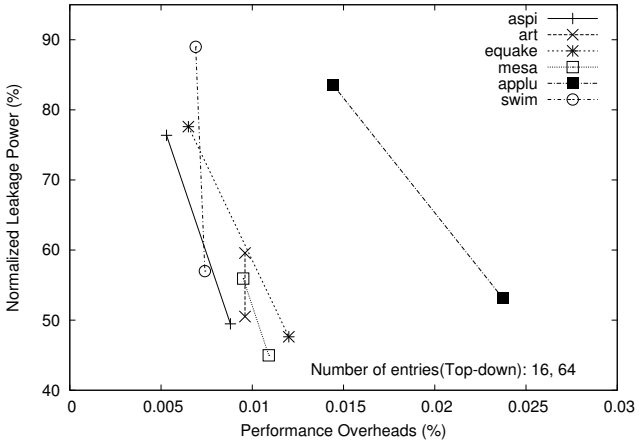


Figure 6.11: Leakage Reduction Efficiency on Data Intensive Applications

leakage reduction effects can be achieved at a cost of mild performance degradation. Since the temporary footprint is closely correlated with the data referencing characteristic of a program rather than the number of dTLB entries, a larger size dTLB means more entries can be put into the low leakage mode in a given slice. Therefore, more leakage power can be saved. On the other hand, a lager dTLB leads to less dTLB misses. The dTLB miss handling process will restore the whole dTLB to the active mode first. If such a situation happens frequently, the leakage reduction opportunity will be diminished, while the performance overheads may be decreased. Since after the 32-entry configuration, further increasing the number of dTLB entries can only introduce a non-distinctive miss ratio reduction, the lines become steeper on the bottom part of the figure. On the whole, a conclusion can be drawn safely that the larger a dTLB is, the better leakage reduction efficiency the proposed mechanism can achieve.

Fig.6.11 shows the normalize leakage power and the performance overheads of a 16-entry and a 64-entry dTLB when applied to 6 floating point programs from the SPEC2000 benchmark [103].

Although embedded processors usually do not support the floating point operation, here, such an evaluation is executed only to verify the robustness of proposed design. By dividing the overall operating time into small time slices, the inherent data referencing locality in and between slices ensures the drowsy opportunity. Thus, leakage reduction can always be achieved for dTLBs with a reasonable size. As shown in the figure, for a 64-entry dTLB, which is the minimum size of contemporary general-purpose processors, about 50% leakage power can be reduced at a cost of 0.04% performance degradation. Notably, proposed mechanism is even applicable to extremely small-sized dTLBs. For instance, the 16-entry dTLB can also achieve an average 26% leakage reduction.

6.5 Related Work

6.5.1 Block Buffering

Previous researches on low-power TLB design are mainly focused on dynamic power. One of the low-power TLB structure is the block buffering [99] [90], where a number of block-buffers are inserted before the main TLB. If two sequential TLB references are located in the same page, the physical address can be generated directly from the block-buffer, without accessing the main TLB. Otherwise, another cycle and power overheads must be paid to access the main TLB. Paper [92] proposed a compiler-based scheme to detect sequential instructions which located on different pages. With such a scheme, only the block-buffer or the main TLB will be accessed, and the overheads caused by block-buffer misses can be avoided. However, the power reduction effects of block buffering TLBs are highly dependent on the spatial locality of TLB references. If the block-buffer hit-rate is low, the power reduction of a block buffering TLB will be degraded significantly. From the perspective of leakage efficient design, leakage reduction techniques may exacerbate the power reduction efficiency with low spatial locality references, due to the performance and power overheads caused by mode transitions. Hence, the block buffering structure is more suitable for instruction TLB instead of data TLB.

6.5.2 Banked TLB

The banked TLB [91] is another low-power TLB structure. It partitions the main TLB into several banks. By accessing only one bank, this technique can effectively reduce the power consumption per TLB access. The drawback of banked TLB is performance degradation due to the tendency to encounter more capacity misses in specific banks. Paper [100] [101] tried to overcome such a drawback by integration the banked TLB and block-buffers. These schemes can selectively access block buffers and TLB banks according to referencing address, and circuit level techniques have also been proposed to remove the comparison latency from the conventional block buffering. As for leakage efficient dTLB design, it may be impossible to satisfy all temporary data referencing requests with only a specific bank. Further, the bank-selective mechanism of banked TLB may not be appropriate for leakage reduction techniques, since banks that stay in low leakage mode can not be restored to the active mode instantaneously.

6.5.3 Drowsy Cache

Another research which is closely related with ours is the drowsy cache [48]. It was proposed to reduce the leakage power of the data cache. By periodically put all cache lines into drowsy mode and active cache lines only when being accessed, the drowsy cache can save the leakage power of a 32KB L1 cache by 52% on average. However, unlike the cache design, the TLB-TAGs occupy a signification portion of the whole TLB. As mentioned in Section 6.3, if TLB-TAGs of all entries are

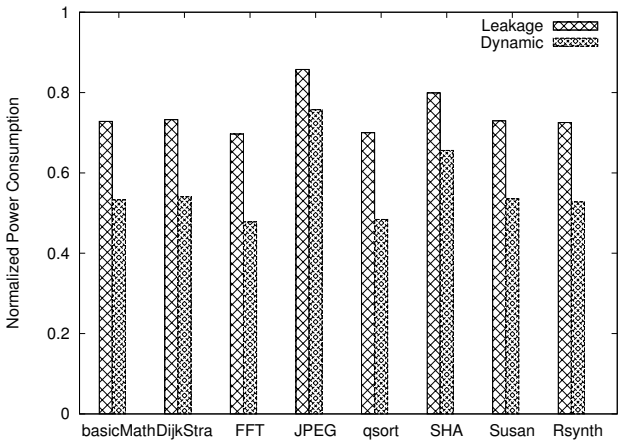


Figure 6.12: Normalized Power Consumption With DS Mechanism: Leakage & Dynamic

activated when a TLB-miss occurs, and such an active state is kept until the next slice, the leakage reduction opportunity will be damaged significantly. Fig.6.12 presents the power reduction effects of a 16-enty dTLB with the DS mechanism which simply adopts the leakage reduction mechanism of drowsy cache to the dTLB design, as mentioned in Section 6.3. The design parameters for power evaluation are the same as mentioned in Section 6.4, and we assume the activation of TLB-TAGs consumes no extra power. As shown in the Fig.6.12, with DS mechanism, leakage power of the dTLB is reduced by 26% on average, and dynamic power is saved by 44%. Comparing with Fig.6.9, our leakage-efficient dTLB design outperforms the one with DS mechanism by 11% in terms of leakage power reduction and 21% for dynamic power reduction.

6.6 Conclusions

A leakage efficient dTLB design has been proposed. By exploiting the dTLB referencing pattern, the proposed design try to fit the temporary footprint of dTLB references by dynamically modifying the number of active dTLB entries. With the integration of the dual voltage supply technique into the dTLB design, entries which are predicted not to be accessed in a given time slice can be put into the low-power mode to save the leakage power consumption. The proposed design also can be employed to reduce dynamic power, with the help of clock gating technique. Evaluation results show 37% of the leakage power and 65% of the dynamic power can be reduced, at a cost of 0.01% performance degradation.

Chapter 7

Conclusions

We have proposed three different leakage-reduction mechanisms on embedded processors. In this chapter, we conclude the whole thesis. In the sections that follow, we summarize the contents of each chapter, as well as views of promising future techniques.

7.1 Summary

As technology scales to deep sub-micron processes, leakage power becomes increasingly significant as compared to dynamic power. Thus, a rethinking of embedded processor design traditionally targeting dynamic power optimization are require in order to reduce the integration cost, enhance the operation reliability, and optimize the battery life for mobile devices. In this thesis, various aspects of leakage-reduction mechanisms on embedded processor have been discussed. Previous chapters can be summarized as follows.

Chapter 1 is an introduction. The motivation, objective, contribution, and organization of this thesis have been introduced.

Chapter 2 presents the background of this thesis. We briefly illustrate the leakage issue in deep sub-micron processes, the classification of leakage power, and leakage power model. Circuit-level leakage-reduction techniques have also been presented. In the last section of this chapter, we present three representative architecture-level leakage-reduction techniques on caches.

Chapter 3 shows a detailed leakage power analysis on embedded processors. For this purpose, we have implemented a MIPS R3000 processor with 65nm CMOS process. The specification of the implemented processor has also been briefly introduced. According to the analysis result, three on-chip components – functional units, instruction TLB and data TLB – have been picked up as the leakage reduction target of this thesis.

Chapter 4 proposes a framework to power gating the functional units at run-time, by integrating circuit-level, architecture-level, and system software techniques. At circuit-level, we propose a fine-grained power gating technique, which has nano-second order wakeup latency and can be implemented at arbitrary granularity. At architecture-level, a PG control scheme, which keeps a functional

unit active only when being used, has been applied. In addition, BET-aware PG control schemes, which are guided by the system software (compiler and operating system) have also been proposed to achieve maximum leakage reduction effects. The proposed mechanisms have been integrated into a real-chip implementation. According to the real-silicon evaluation results, the leakage power consumption of the processor core can be reduced by 5% at 25℃and 23% at 80℃.

Chapter 5 presents a leakage-efficient instruction TLB design. The key observation is that when programs enter a physical page, the following instructions tend to be fetched from the same page for a rather long time. Thus, by employing a small storage component which holds the recent address translation information, the TLB access frequency can be drastically decreased, and the instruction TLB can be turned into the low-leakage mode with the dual voltage supply technique. Based on such a design philosophy, three leakage control policies are proposed to maximize the leakage reduction efficiency. Evaluation results with eight MiBench programs show that the proposed design can reduce the leakage power of the instruction TLB by 50% on average, with only 0.01% performance degradation.

Chapter 6 presents a leakage efficient data TLB design. Due to the data locality embodied in programs, data TLB references tend to hit only a small number of pages during short execution intervals. After dividing the overall execution time into smaller time slices, a leakage reduction mechanism is proposed to detect TLB entries which actually serve for virtual-to-physical address translations within each time slice. Thus, with the integration of the dual voltage supply technique, those TLB entries which are not used for address translations can be put into low leakage mode to save power. Evaluation results with eight MiBench programs show that the proposed design can reduce the leakage power of a data TLB by 37% on average, with performance degradation less than 0.01%

7.2 Future Directions

7.2.1 Geyser-2

Since Geyser-1 does not include on-chip caches, its maximum working frequency is restricted to 60MHz. We have assumed the wakeup latency of fine-grained power gating is one clock cycle, thus, the real-chip evaluation can only verify that the wakeup latency is less than 17ns, although circuit-level simulation shows it is less than 5ns. Another problem is that the operating system is difficult to be ported without on-chip TLB. To address these problems, a new prototype chip – Geyser-2, which is designed with on-chip caches and TLB, has been fabricated. A preliminary real-chip evaluation shows the fine-grained PG can work at 210MHz without incurring any electric problems. Evaluating the leakage reduction effects of Geyser-2 with benchmark programs will be our next-step work.

7.2.2 Fine-grained Power Gating with UPF

In chapter 4, we have set up the design flow for fine-grained PG, and have verified its feasibility with real-silicon. The proposed design flow employs the Locally-Shared Virtual ground scheme, and the power switches can be distributed arbitrarily without constraints imposed by permanent power network connection. Although such scheme achieves high speed wake up and reasonable leakage reduction, it needs customize standard cells for making a virtual ground pin inside each cell with precise characterization data.

Recent Synopsys's UPF [87] begin to support a column-based PG method. Although the distribution of the power switches is confined beneath the power straps, similar leakage reduction effects and wakeup latency may be achieved when column-based PG method is appropriately applied without the necessity to generate a special set of cells. However, such a method has not been reported in literature for functional units design. A comparative study on our fine-grained PG method and the one supported by UPF will be an interesting research topic.

7.2.3 Reducing the Power Consumption of TLBs on Chip Multiprocessors

Nowadays, the focus of microprocessor design has been shifted to chip multiprocessors (CMPs), even in the field of embedded applications. Study on the leakage-reduction mechanisms of TLBs on chip-multiprocessors will be another interesting research topic. Interaction with other processor cores may change the access pattern of TLBs significantly. In paper [104], authors propose a mechanism to using virtual cache to filter out unnecessary TLB and snoop power consumption. Another potential mechanism to reduce the access frequency of TLBs may be achieved by integrated the bloom filter [105]. With such a mechanism, the TLB which actually contain the virtual address of the broadcast signal will be accessed.

Bibliography

[1] S. Borkar. "Design challenges of technology scaling". *IEEE micro*, 19(4):23–29, 1999.

[2] V. Venkatachalam and M. Franz. "Power reduction techniques for microprocessor systems". *ACM Computing Surveys (CSUR)*, 37(3):195–237, 2005.

[3] T. Mudge. "Power: a first-class architectural design constraint". *Computer*, 34(4):52–58, 2001.

[4] R. Wilson and D. Lammers. "Grove Calls Leakage Chip Designers' Top Problem". *Electronic Engineering Times*, page 80, 2002.

[5] N.S. Kim, T. Austin, D. Baauw, T. Mudge, K. Flautner, J.S. Hu, M.J. Irwin, M. Kandemir, and V. Narayanan. "Leakage current: Moore's law meets static power". *Computer*, 36(12):68–75, 2003.

[6] ITRS. Int'l Technology Roadmap for Semiconductor, http://public.itrs.net, 2001.

[7] B. Doyle, R. Arghavani, D. Barlage, S. Datta, M. Doczy, J. Kavalieros, A. Murthy, and R. Chau. "Transistor elements for 30nm physical gate lengths and beyond". *Intel Technology Journal*, 6(2):42–54, 2002.

[8] Erin Farquhar and Philip Bunce. *THE MIPS PROGRAMMER'S HANDBOOK*. Morgan Kaufmann Publishers, 1994.

[9] M.T. Bohr, R.S. Chau, T. Ghani, and K. Mistry. "The high-k solution". *IEEE Spectrum*, 44(10):29–35, 2007.

[10] S. Kaxiras and M. Martonosi. "Computer Architecture Techniques for Power-Efficiency". *Synthesis Lectures on Computer Architecture*, 3(1):1–207, 2008.

[11] T. Sakurai and A.R. Newton. "Alpha-power law MOSFET model and its applications to CMOS inverter delay and other formulas". *IEEE Journal of Solid-State Circuits*, 25(2):584–594, 2002.

[12] P. Ko, J. Huang, Z. Liu, and C. Hu. "BSIM3 for analog and digital circuit simulation". In *Proceedings of the IEEE Symposium on VLSI Technology CAD*, pages 400–429, 1993.

[13] J.A. Butts and G.S. Sohi. "A static power model for architects". In *Proceedings of the 33rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-33)*, pages 191–201. IEEE, 2002.

[14] S. Narendra, V. De, D. Antoniadis, A. Chandrakasan, and S. Borkar. "Scaling of stack effect and its application for leakage reduction". In *Proceedings of the 2001 international symposium on Low power electronics and design*, pages 195–200. ACM, 2001.

[15] Y. Ye, J. Tschanz, S. Narendra, S. Borkar, M. Stan, and V. De. "Comparative delay, noise and energy of high-performance domino adders with stack node preconditioning (SNP)". In *Digest of Technical Papers of 2000 Symposium on VLSI Circuits*, pages 188–191. IEEE, 2000.

[16] L.T. Clark, N. Deutscher, S. Demmons, and F. Ricci. "Standby power management for a 0.18μm microprocessor". In *Proceedings of the 2002 international symposium on Low power electronics and design*, pages 7–12. ACM, 2002.

[17] H. Ananthan, C.H. Kim, and K. Roy. "Larger-than-vdd forward body bias in sub-0.5 V nanoscale CMOS". In *Proceedings of the 2004 International Symposium on Low Power Electronics and Design*, pages 8–13. IEEE, 2004.

[18] M. Anis, S. Areibi, M. Mahmoud, and M. Elmasry. "Dynamic and leakage power reduction in MTCMOS circuits using an automated efficient gate clustering technique". In *Proceedings of Design Automation Conference 2002*, pages 480–485. ACM, 2002.

[19] S. Mutoh, T. Douseki, Y. Matsuya, T. Aoki, S. Shigematsu, and J. Yamada. "1-V power supply high-speed digital circuit technology with multithreshold-voltage CMOS". *IEEE Journal of Solid-State Circuits*, 30(8):847–854, 2002.

[20] T. Kuroda, T. Fujita, S. Mita, T. Nagamatsu, S. Yoshioka, K. Suzuki, F. Sano, M. Norishima, M. Murota, M. Kako, and M. Kinugawa. "A 0.9-V, 150-MHz, 10-mW, 4 mm2, 2-D discrete cosine transform core processor with variable threshold-voltage (VT) scheme". *IEEE Journal of Solid-State Circuits*, 31(11):1770–1779, 1996.

[21] L. Wei, Z. Chen, M. Johnson, K. Roy, and V. De. "Design and optimization of low voltage high performance dual threshold CMOS circuits". In *Proceedings of the 35th annual Design Automation Conference*, pages 489–494. ACM, 1998.

[22] R.S. Cobbold. "Temperature effects in MOS transistors". *Electronics Letters*, 2(6):190–191, 1966.

[23] E.A. Gutiérrez-D, M.J. Deen, and C.L. Claeys. *Low temperature electronics: physics, devices, circuits, and applications*. Academic Pr, 2001.

[24] Y. Tsividis. *Operation and Modeling of the MOS Transistor*. McGraw-Hill, Inc., 1987.

[25] L. Vadasz and A.S. Grove. "Temperature dependence of MOS transistor characteristics below saturation". *IEEE Transactions on Electron Devices*, 13(12):863–866, 1966.

[26] L. He, W. Liao, and M.R. Stan. "System level leakage reduction considering the interdependence of temperature and leakage". In *Proceedings of the 41st annual Design Automation Conference*, pages 12–17. ACM, 2004.

[27] F. Hamzaoglu and M.R. Stan. "Circuit-level techniques to control gate leakage for sub-100nm CMOS". In *Proceedings of the 2002 international symposium on Low power electronics and design*, pages 60–63. ACM, 2002.

[28] S. Song, H. Kim, J.Y. Yoo, J.H. Yi, W.S. Kim, N.I. Lee, K. Fujihara, H.K. Kang, and J.T. Moon. "On the gate oxide scaling of high performance CMOS transistors". In *IEDM Technical Digest of 2001 International Electron Devices Meeting*, pages 321–324. IEEE, 2001.

[29] R.S. Chau. "Intel's Breakthrough in High-K Gate Dielectric Drives Moore's Law Well into the Future". *Intel Technology Magazine*, pages 1–7, 2004.

[30] R.M. Rao, J.L. Burns, A. Devgan, and R.B. Brown. "Efficient techniques for gate leakage estimation". In *Proceedings of the 2003 international symposium on Low power electronics and design*, pages 100–103. ACM, 2003.

[31] A. Keshavarzi, K. Roy, and C.F. Hawkins. "Intrinsic leakage in low power deep submicron CMOS ICs". In *Proceedings of 1997 International Test Conference*, pages 146–155. IEEE, 1997.

[32] A. Keshavarzi, S. Ma, S. Narendra, B. Bloechel, K. Mistry, T. Ghani, S. Borkar, and V. De. "Effectiveness of reverse body bias for leakage control in scaled dual Vt CMOS ICs". In *Proceedings of 2001 International Symposium on Low Power Electronics and Design*, pages 207–212. IEEE, 2001.

[33] K. Von Arnim, E. Borinski, P. Seegebrecht, H. Fiedler, R. Brederlow, R. Thewes, J. Berthold, and C. Pacha. "Efficiency of body biasing in 90-nm CMOS for low-power digital circuits". *IEEE Journal of Solid-State Circuits*, 40(7):1549–1556, 2005.

[34] D. Chinnery and K.W. Keutzer. *Closing the power gap between ASIC & custom: tools and techniques for low power design*. Springer-Verlag New York Inc, 2007.

[35] B. Calhoun, J. Kao, and A. Chandrakasan. "Power gating and dynamic voltage scaling". *Nanometer CMOS Technologies*, pages 41–75, 2006.

[36] T. Mudge, K. Flautner, D. Blaauw, and S.M. Martin. "Combined dynamic voltage scaling and adaptive body biasing for lower power microprocessors under dynamic workloads". In *Proceedings of International Conference on Computer-Aided Design (ICCAD2002)*, pages 721–725. IEEE/ACM, 2002.

[37] S. Sirichotiyakul, T. Edwards, C. Oh, J. Zuo, A. Dharchoudhury, R. Panda, and D. Blaauw. "Stand-by power minimization through simultaneous threshold voltage selection and circuit sizing". In *Proceedings of the 36th annual ACM/IEEE Design Automation Conference*, pages 436–441. ACM, 1999.

[38] L. Wei, Z. Chen, M. Johnson, K. Roy, and V. De. "Design and optimization of low voltage high performance dual threshold CMOS circuits". In *Proceedings of the 35th annual Design Automation Conference*, pages 489–494. ACM, 1998.

[39] Q. Wang and S.B.K. Vrudhula. "Algorithms for minimizing standby power in deep submicrometer, dual-Vt CMOS circuits". *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 21(3):306–318, 2002.

[40] K. Usami, N. Kawabe, M. Koizumi, K. Seta, and T. Furusawa. "Automated selective multi-threshold design for ultra-low standby applications". In *Proceedings of the 2002 international symposium on Low power electronics and design*, pages 202–206. ACM, 2002.

[41] S. Mutoh, T. Douseki, Y. Matsuya, T. Aoki, S. Shigematsu, and J. Yamada. "1-V power supply high-speed digital circuit technology with multithreshold-voltage CMOS". *IEEE Journal of Solid-State Circuits*, 30(8):847–854, 1995.

[42] V. De and S. Borkar. "Technology and design challenges for low power and high performance". In *Proceedings of the 1999 international symposium on Low power electronics and design*, pages 163–168. ACM, 1999.

[43] K. Shi and D. Howard. "Challenges in sleep transistor design and implementation in low-power designs". In *Proceedings of the 43rd annual Design Automation Conference*, pages 113–116. ACM, 2006.

[44] H. Kawaguchi, K. Nose, and T. Sakurai. "A super cut-off CMOS (SCCMOS) scheme for 0.5-V supply voltage with picoampere stand-by current". *IEEE Journal of Solid-State Circuits*, 35(10):1498–1501, 2002.

[45] K.S. Min, H. Kawaguchi, and T. Sakurai. "Zigzag super cut-off CMOS (ZSCCMOS) block activation with self-adaptive voltage level controller: An alternative to clock-gating scheme in leakage dominant era". In *Digest of Technical Papers of 2003 IEEE International Solid-State Circuits Conference*, pages 400–502. IEEE, 2005.

[46] Y. Shin, J. Seomun, K.M. Choi, and T. Sakurai. "Power gating: Circuits, design methodologies, and best practice for standard-cell VLSI designs". *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 15(4):28, 2010.

[47] S. Kaxiras, Z. Hu, and M. Martonosi. "Cache decay: exploiting generational behavior to reduce cache leakage power". *Proceedings of the 28th annual international symposium on Computer architecture*, pages 240–251, 2001.

[48] 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*, 12(2):167–184, 2004.

[49] S.W. Chung and K. Skadron. "On-Demand Solution to Minimize I-Cache Leakage Energy with Maintaining Performance". *Transactions on Computers*, 57(1):7–24, 2008.

[50] C.L. Yang and C.H. Lee. "HotSpot cache: joint temporal and spatial locality exploitation for i-cache energy reduction". In *Proceedings of the 2004 International Symposium on Low Power Electronics and Design*, pages 114–119. IEEE, 2004.

[51] M. Powell, S.H. Yang, B. Falsafi, K. Roy, and TN Vijaykumar. "Gated-V dd: a circuit technique to reduce leakage in deep-submicron cache memories". In *Proceedings of the 2000 international symposium on Low power electronics and design*, pages 90–95. ACM, 2000.

[52] S.H. Yang, MD Powell, B. Falsafi, and T.N. Vijaykumar. "Exploiting choice in resizable cache design to optimize deep-submicron processor energy-delay". In *Proceedings of 8th International Symposium on High-Performance Computer Architecture, 2002*, pages 151–161, 2002.

[53] W. Zhang, JS Hu, V. Degalahal, M. Kandemir, N. Vijaykrishnan, and M.J. Irwin. "Compiler-directed instruction cache leakage optimization". In *Proceedings of 35th Annual IEEE/ACM International Symposium on Microarchitecture, 2002 (MICRO-35)*, pages 208–218. IEEE, 2002.

[54] Y.J. Chen, C.L. Yang, J.W. Chi, and J.J. Chen. "TACLC: Timing-Aware Cache Leakage Control for Hard Real-Time Systems". *IEEE Transactions on Computers*, 2011.

[55] D. Kudithipudi, S. Petko, and E. John. "Cache leakage power analysis in embedded applications". In *The 47th Midwest Symposium on Circuits and Systems, 2004 (MWSCAS'04)*, volume 2, pages II–517. IEEE, 2004.

[56] N.S. Kim, D. Blaauw, and T. Mudge. "Quantitative analysis and optimization techniques for on-chip cache leakage power". *IEEE Transactions on Very Large Scale Integration Systems*, 13(10):1147–1156, 2005.

[57] W. Zhang. "Exploiting loop behavior for data cache leakage reduction". *Journal of Embedded Computing*, 1(4):501–508, 2005.

[58] M. Goudarzi and T. Ishihara. "Instruction cache leakage reduction by changing register operands and using asymmetric sram cells". In *Proceedings of the 18th ACM Great Lakes symposium on VLSI*, pages 383–386. ACM, 2008.

[59] H. Homayoun, M. Makhzan, and A. Veidenbaum. "Multiple sleep mode leakage control for cache peripheral circuits in embedded processors". In *Proceedings of the 2008 international conference on Compilers, architectures and synthesis for embedded systems*, pages 197–206. ACM, 2008.

[60] D.A. Wood, M.D. Hill, and RE Kessler. "A model for estimating trace-sample miss ratios". *ACM SIGMETRICS Performance Evaluation Review*, 19(1):79–89, 1991.

[61] H. Hanson, M.S. Hrishikesh, V. Agarwal, S.W. Keckler, and D. Burger. "Static energy reduction techniques for microprocessor caches". *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 11(3):303–313, 2003.

[62] J.K. Peir, S.C. Lai, S.L. Lu, J. Stark, and K. Lai. "Bloom filtering cache misses for accurate data speculation and prefetching". In *Proceedings of the 16th international Conference on Supercomputing*, pages 189–198. ACM, 2002.

[63] S. Petit, J. Sahuquillo, J.M. Such, and D. Kaeli. "Exploiting temporal locality in drowsy cache policies". In *Proceedings of the 2nd conference on Computing frontiers*, pages 371–377. ACM, 2005.

[64] M. Schlett. "Trends in embedded-microprocessor design". *Computer*, 31(8):44–49, 1998.

[65] D. Sweetman. *See MIPS Run*. Morgan Kaufmann, 2006.

[66] S. Cotofana, S. Wong, S. Vassiliadis, and S. Stamatis. "Embedded processors: characteristics and trends". In *Proceedings of the 2001 ASCI Conference*, pages 17–24. Citeseer, 2001.

[67] J.L. Hennessy and D.A. Patterson. *Computer Architecture: A Quantitative Approach, 4th edition*. Morgan Kaufmann, 2003.

[68] M.D. Hill and A.J. Smith. "Experimental evaluation of on-chip microprocessor cache memories". *ACM SIGARCH Computer Architecture News*, 12(3):158–166, 1984.

[69] SYNOPSYS. http://www.synopsys.com.

[70] Inc. Mentor. http://www.mentor.com.

[71] M.R. Guthaus, J.S. Ringenberg, D. Ernst, M.T. Austin, T. Mudge, and B.R. Brown. "MiBench: A free, commercially representative embedded benchmark suite". In *IEEE International Workshop on Workload Characterization, 2001. WWC-4. 2001*, pages 3–14, 2001.

[72] R. Jotwani, S. Sundaram, S. Kosonocky, A. Schaefer, V. Andrade, G. Constant, A. Novak, and S. Naffziger. "An x86-64 core implemented in 32nm SOI CMOS". In *Proceedings 2010 IEEE International Solid-State Circuits Conference (ISSCC)*, pages 106–107. IEEE, 2010.

[73] R. Kumar and G. Hinton. "A family of 45nm IA processors". In *Digest of Technical Papers of 2009 IEEE International Solid-State Circuits Conference*, pages 58–59. IEEE, 2009.

[74] Y. Kanno. "Hierarchical power distribution with 20 power domains in 90-nm low-power multi-CPU processor". *Digest of Technical Papers of 2006 IEEE International Solid-State Circuits Conference*, pages 540–541, February 2006.

[75] Z. Hu, A. Buyuktosynoglu, V. Srinivasan, V. Zyuban, H. Jacobson, and P. Bose. "Microarchitectural techniques for power gating of execution units". In *Proceedings of the 2006 International Symposium on Low Power Electronics and Design*, pages 32–37, August 2006.

[76] A. Youssef, M. Anis, and M. Elmasry. "Dynamic standby prediction for leakage tolerant microprocessor functional units". In *39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'06)*. IEEE Computer Society, 2006.

[77] A. Lungu, A. Buyuktosunoglu, and D.J. Sorin. "Dynamic power gating with quality guarantees". In *Proceedings of the 14th ACM/IEEE international symposium on Low power electronics and design*, pages 377–382. ACM New York, NY, USA, 2009.

[78] W. Zhang, M. Kandemir, N. Vijaykrishnan, MJ Irwin, and V. De. "Compiler support for reducing leakage energy consumption". In *Proceedings of 2005 Europe Conference and Exhibition on Design, Automation and Test*, pages 1146–1147. IEEE, 2005.

[79] S. Rele, S. Pande, S. Onder, and R. Gupta. "Optimizing static power dissipation by functional units in superscalar processors". In *Proceedings of Compiler Construction*, pages 85–100. Springer, 2002.

[80] S. Roy, N. Ranganathan, and S. Katkoori. "A Framework for Power-Gating Functional Units in Embedded Microprocessors". *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 17(11):1640–1649, 2009.

[81] N. Seki, Z. Lei, Y. Kojima, D. Ikebuchi, Y. Hasegawa, N. Ohkubo, S. Takeda, T. Kashima, T. Shirai, K. Usami, T. Sunata, J. Kanai, M Mitara, M. Kondo, H. Nakamura, and H. Amano. "A fine grain dynamic sleep control scheme in MIPS R3000". In *Proceedings of 26th International Conference on Computer Design*, pages 612–617. IEEE, 2008.

[82] N.Seki, Z.Lei, Y.Kojima, D.Ikebuchi, Y.Hasegawa, N.Ohkubo, S.Takeda, T.Kashima, T.Shirai, K.Usami, T.Sunata, J.Kanai, M.Mitara, H.Kondo, H.Nakamura, and H.Amano. "A fine grain dynamic sleep control scheme in MIPS R3000". *IEICE Transactions on Information and System (Japanese Edition)*, J93(6):920–930, 2010.

[83] D. Ikebuchi, N. Seki, Y. Kojima, M. Kamata, Z. Lei, H. Amano, T. Shirai, S. Koyama, T. Hashida, Y. Umahashi, et al. "Geyser-1: a MIPS R3000 CPU core with fine grain runtime power gating". In *Proceedings of 2009 IEEE Asian Solid-State Circuits Conference*, pages 281–284. IEEE, 2009.

[84] Z.Lei, D.Ikebuchi, Y.Saito, M.Kamata, N.Seki, Y.Kojima, H.Amano, S.Koyama, T.Hashida, Y.Umahashi, D.Masuda, K.Usami, T.Sunata, K.Kimura, and S.Takeda M.Namiki, H.Nakamura, and M.Kondo. "Geyser-1 and Geyser-2: MIPS R3000 CPU chips with fine-grain runtime Power gating". In *13rd IEEE Symposium on Low-Power and High-Speed Chips*, pages 161–163. IEEE, 2010.

[85] K. Usami and N. Ohkubo. "An approach for fine-grained run-time power gating using locally extracted sleep signals". In *Proceedings of 24th International Conference on Computer Design*, pages 155–161. IEEE, 2006.

[86] K. Usami, T. Shirai, T. Hashida, H. Masuda, S. Takeda, M. Nakata, N. Seki, H. Amano, M. Namiki, M. Imai, et al. "Design and implementation of fine-grain power gating with ground bounce suppression". In *Proceedings of 22nd International Conference on VLSI Design, 2009*, pages 381–386. IEEE, 2009.

[87] IEEE Computer Society. "IEEE Standard for Design and Verification of Low Power Integrated Circuits". In *IEEESTD.2009.4809845*, March 2009.

[88] M. Keating, D. Flynn, R. Aitken, A. Gibbons, and K. Shi. *Low power methodology manual: for system-on-chip design*. Springer Verlag, 2007.

[89] Inc. Sequence Design. http://www.sequencedesign.com.

[90] L.T. Clark, B. Choi, and M. Wilkerson. "Reducing translation lookaside buffer active power". In *Proceedings of the 2003 international symposium on Low power electronics and design*, pages 10–13. ACM Press New York, NY, USA, 2003.

[91] Y.J. Chang. "An ultra low-power TLB design". In *Proceedings of the 2006 conference on Design, automation and test in Europe*, pages 1122–1127. IEEE, 2006.

[92] I. Kadayif, A. Sivasubramaniam, M. Kandemir, G. Kandiraju, and G. Chen. "Optimizing instruction TLB energy using software and hardware techniques". *ACM Trans. Des. Autom. Electron. Syst.*, 10(2):229–257, 2005.

[93] QEMU. http://fabrice.bellard.free.fr/qemu.

[94] K. Matsuo, M. Sato, and M. Namiki. "Development of system evaluation environment using QEMU for low power system". In *Proceedings of Symposium on Advanced Computing Systems and Infrastructures*, pages 61–68, 2007.

[95] V. Packirisamy, A. Zhai, W.C. Hsu, P.C. Yew, and T.F. Ngai. "Exploring speculative parallelism in SPEC2006". In *Proceedings of Performance Analysis of Systems and Software, 2009. ISPASS 2009. IEEE International Symposium on*, pages 77–88. IEEE, 2009.

[96] G.H. Loh, S. Subramaniam, and Y. Xie. "Zesto: A cycle-level simulator for highly detailed microarchitecture exploration". In *Proceedings of IEEE International Symposium on Performance Analysis of Systems and Software, 2009. ISPASS 2009.*, pages 53–64. IEEE, 2009.

[97] R. Giorgi and P. Bennati. "Reducing leakage in power-saving capable caches for embedded systems by using a filter cache". In *Proceedings of the 2007 workshop on MEmory performance: DEaling with Applications, systems and architecture*, pages 97–104. ACM, 2007.

[98] M. Zhang and N.R. Shanbhag. "Soft-error-rate-analysis (SERA) methodology". *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(10):2140–2155, 2006.

[99] D. Fan, Z. Tang, H. Huang, and G.R. Gao. "An energy efficient TLB design methodology". In *Proceedings of the 2005 International Symposium on Low Power Electronics and Design*, pages 351–356, 2005.

[100] Y.J. Chang and M.F. Lan. "Two new techniques integrated for energy-efficient TLB design". *IEEE transactions on very large scale integration (VLSI) systems*, 15(1):13–23, 2007.

[101] J.H. Lee, G.H. Park, S.B. Park, and S.D. Kim. "A selective filter-bank TLB system". In *Proceedings of the 2003 international symposium on Low power electronics and design*, pages 312–317. ACM, 2003.

[102] G.L. Steele. *Common LISP: the language*. Digital Pr, 1990.

[103] SPEC2000. http://www.spec.org.

[104] M. Ekman, P. Stenstrom, and F. Dahlgren. "TLB and snoop energy-reduction using virtual caches in low-power chip-multiprocessors". In *Proceedings of the 2002 international symposium on Low power electronics and design*, pages 243–246. ACM, 2002.

[105] J.K. Peir, S.C. Lai, S.L. Lu, J. Stark, and K. Lai. "Bloom filtering cache misses for accurate data speculation and prefetching". In *Proceedings of the 16th international conference on Supercomputing*, pages 189–198. ACM, 2002.

Publications

Journal Papers

[1] Zhao Lei, Hui Xu, Daisuke Ikebuchi, Tetsuya Sunata, Mitaro Namiki, Hideharu Amano. "A Leakage Efficient Data TLB Design for Embedded Processors". *IEICE Transactions on Information and Systems, Vol. E94-D, No.1*, pp.51-59, Jun. 2011.

[2] Zhao Lei, Hui Xu, Daisuke Ikebuchi, Tetsuya Sunata, Mitaro Namiki, Hideharu Amano. "A Leakage Efficient Instruction TLB Design for Embedded Processors". *IEICE Transactions on Information and Systems, Vol. E94-D, No.8*, Aug. 2011. *in press*

[3] Zhao Lei, Daisuke Ikebuchi, Kimiyoshi Usami, Mitaro Namiki, Masaaki Kondo, Hiroshi Nakamura, Hideharu Amano. "Design and Implementation Fine-grained Power Gating on Microprocessor Functional Units". *IPSJ Transactions on System LSI Design Methodology, Vol.4 August Issue*, Aug. 2011. *in press*

[4] Naomi Seki, Zhao Lei, Yu Kojima, Daisuke Ikebuchi, Yohei Hasegawa, Naoaki Ohkubo, Seidai Takeda, Toshihiro Kashima, Toshiaki Shirai, Kimiyoshi Usami, Tetsuya Sunata, Jun Kanai, Mitaro Namiki, Hiroaki Konda, Hiroshi Nakamura, and Hideharu Amano. "A Fine Grain Dynamic Sleep Control Scheme in MIPS R3000". *IEICE Transaction on Information and System, Vol. J93-D, No.6*, pp 575-584, Jun. 2010. (In Japanese)

International Conference Papers

[5] Zhao Lei, Daisuke Ikebuchi, Yoshiki Saito, Masahiro Kamata, Naomi Seki, Yu Kojima, Hideharu Amano, Satoshi Koyama, Tatsunori Hashida, Yusuke Umahashi, Daiki Masuda, Kimiyoshi Usami, Kazuki Kimura, Mitaro Namiki, Seidai Takeda, Hiroshi Nakamura, Masaaki Kondo. "Geyser-2: The Second Prototype CPU with Fine-grained Run-time Power Gating". *Proc. of 16th Asia and South Pacific Design Automation Conference (ASP-DAC2011)*, pp.87-88, Jan. 2011.

[6] Zhao Lei, Hui Xu, Dasuke Ikebuchi, Hideharu Amano, Tetsuya Sunata, Mitaro Namiki. "Reducing instruction TLB's leakage power consumption for embedded processors." *Proc. of 1st International Conference on Green Computing (GreenComp2010)*, pp.477-484, August 2010.

[7] Z. Lei, D. Ikebuchi, Y. Saito, M. Kamata, N. Seki, Y. Kojima, H. Amano, S. Koyama, T. Hashida, Y. Umahashi, D. Masuda, K. Usami, T. Sunata, K. Kimura, M. Namiki, S. Takeda, H. Nakamura and M. Kondo. "Geyser-1 and Geyser-2: MIPS R3000 CPU Chips with Fine-grain Runtime Power Gating". *Proc. of 13rd IEEE Symposium on Low-Power and High-Speed Chips (Cool Chips 2010)*, pp.457-459, Apr. 2010.

[8] Zhao Lei, Hui Xu, Nanomi Seki, Saito Yoshiki, Yohei Hasegawa, Kimiyoshi Usami, and Hideharu Amano. "Cache Controller Design on Ultra Low Leakage Embedded Processors". *Proc. of 22nd International Conference on Architecture of Computing Systems (ARCS2009)*, pp.171-182, March 2009.

[9] Zhao Lei, Hui Xu, Nanomi Seki, Saito Yoshiki, Yohei Hasegawa, Kimiyoshi Usami and Hideharu Amano. "A Run-time Power Gating Sleep Control Scheme on Cache Controller". *Proc. of 11th IEEE Symposium on Low-Power and High-Speed Chips (Cool Chips 2008)*, Poster Session, Apr. 2008.

[10] D. Ikebuchi, Y. Saito, M. Kamata, N. Seki, Y. Kojima, Z. Lei, H. Amano, S. Koyama, T. Hashida, Y. Umahashi, D. Masuda, K. Usami, T. Sunata, K. Kimura, M. Namiki, S. Takeda, H. Nakamura and M. Kondo. "Geyser-1: A MIPS R3000 CPU core with Fine Grain Runtime Power Gating". *Proc. of IEEE Asian Solid-State Circuits Conference 2009 (A-SSCC2009)*, pp.16-18, Nov. 2009.

[11] N. Seki, Z. Lei, J. Kei, D. Ikebuchi, Y. Kojima, Y. Hasegawa, H. Amano, T. Tashima, S. Takeda, T. Hirai, M. Nakata, K. Usami, T. Sunata, J. Kanai, M. Namiki, M. Kondo, and H. Nakamura. "A Fine-grain Dynamic Sleep Control Scheme in MIPS R3000". *Proc. of 26th International Conference on Computer Design (ICCD2008)*, pp.162-167, Oct. 2008.

Domestic Conference Papers and Technical Reports

[12] Zhao Lei, Hui Xu, Daisuke Ikebuchi, Kamata, Mitaro Namiki, Hideharu Amano. "Leakage Efficient TLB Design for Embedded Processors". *IPSJ SIG Technical Reports (SWoPP09), Vol. 184, No. 13,* CD-ROM9 pages, Aug. 2009.

[13] Zhao Lei, Hui Xu, Nanomi Seki, Saito Yoshiki, Yohei Hasegawa, Kimiyoshi Usami and Hideharu Amano. "Cache Controller Design with Run-time Power Gating". *IPSJ SIG Technical Reports (SWoPP08), Vol. 2008, No. 75,* pp.127-132, Aug. 2008.

[14] H. Xu, Z. Lei, S. Naomi, H. Amano, M. Namiki, K. Usami. "Reducing Power of TLB with Power-Gating Technique on Microprocessor". *IPSJ SIG Technical Reports (SWoPP08), Vol. 2008, No. 75,* pp.121-126, Aug. 2008.

[15] Nanomi Seki, Lei Zhao, Xu Hui, Yohei Hasegawa, Hideharu Amano, Kimiyoshi Usami, Mitaro Namiki, Nakamura Hiroi. "A Fine-grain Dynamic Sleep Control Scheme in MIPS

R3000". *Proc. of the 6th Symposium on Advanced Computing Systems and Infrastructurea (SACSIS'08)*, pp.201-209, Jun. 2008. (In Japanese)