PAPER

A Leakage Efficient Instruction TLB Design for Embedded Processors

Zhao LEI[†a)], **Hui XU**[†], **Daisuke IKEBUCHI**[†], **Tetsuya SUNATA**[††], ***Nonmembers***, **Mitaro NAMIKI**[††], ***and*** **Hideharu AMANO**[†], ***Members***

SUMMARY This paper presents a leakage-efficient instruction TLB (Translation Lookaside Buffer) design for embedded processors. 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.

key words: *leakage power, TLB, embedded processor*

1. Introduction

Power 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 scaling. Now, reducing 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 and has become an active research field since the last decade. As for processors, previous leakage reduction mechanisms [1]–[4] were mainly applied to less active components (multipliers and dividers in function units) or partially utilized components (the cache memory). With integration of circuit-level low-leakage techniques, those mechanisms can optimize the leakage power of a selected target by appropriately switching it between the low-leakage mode and the active mode. However, to ensure the overall leakage reduction effect, leakage reduction mechanisms on other components are also necessary.

Manuscript received October 28, 2010.
Manuscript revised March 7, 2011.

[†]The authors are with the Faculty of Science and Technology, Keio University, Yokohama-shi, 223–0061 Japan.
[††]The authors are with the Department of Computer and Information Sciences, Tokyo University of Agriculture and Technology, Koganei-shi, 184–8588 Japan.
a) E-mail: lei@am.ics.keio.ac.jp

DOI: 10.1587/transinf.E94.D.1565

The translation Look-aside Buffer (TLB) is an important component even 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 Geyser-0 [4], a 16-entry TLB consumes as much as 38% of dynamic and 29% of leakage power when compared to a MIPS R3000 processor core [*]. Many publications have been devoted to exploring the dynamic power reduction mechanisms on TLBs [5]–[7], 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 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 [7], 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 paper 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 [1] and drowsy cache [2], 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. Moreover, the iTLB is one of the most active components in embedded processors with a high utilization, which intuitively does not leave much room 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 considerably long time. Hence, by inserting a small size storage component, which keeps the recent address-translation information, between the proces-

[*]The processor core does not include on-chip cache. Both the TLB and the processor core are implemented with 90 nm CMOS technology, and the TLB is a mixed instruction/data one.

[**]Since the data TLB has a very different referencing pattern, its leakage reduction mechanism will be another work.

sor 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 a uniform low power iTLB solution.

The remainder of this paper is organized as follows. The next section presents a leakage-efficient iTLB structure based on the locality analysis of iTLB references. In Sect. 3, detailed leakage control policies and hardware implementation will be illustrated. Leakage saving results will be shown in Sect. 4, and we will discuss the related work in Sect. 5. Section 6 concludes the paper.

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 may directly degrade the maximum clock 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.

Table 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 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

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 [8]. 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 [9] are executed. A group of authors [10] modified the basic structure of QEMU, so that it can be used to trace both TLB references and TLB-flush information. Table 1 shows the configuration parameters of the emulator, and the number of TLB-flushes of each application program is shown in Table 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.

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 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. Figure 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

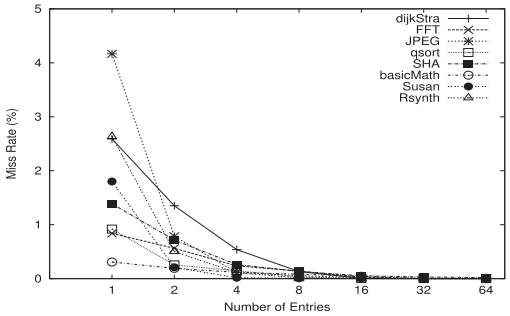


Fig. 1 iTLB miss rate.

low miss rate can be observed from 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 about 2%. This 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.

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.

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. Figure 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

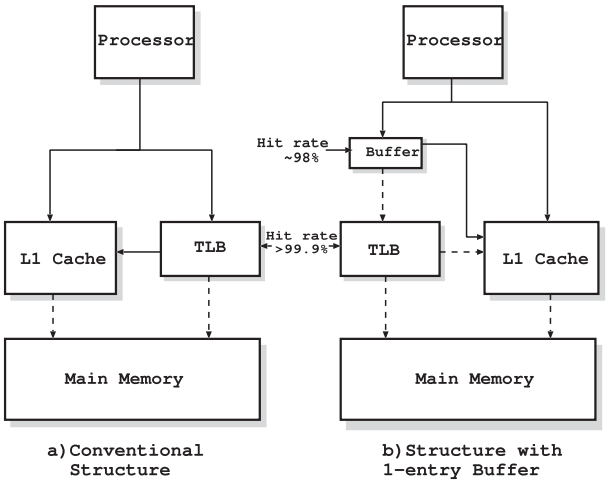


Fig. 2 Structure of the conventional iTLB and the leakage efficient iTLB.

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 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 Sect. 3.2). Comparing with the structure using entry/line granularity leakage control [2], [11], proposed one is more suitable for the IP-reuse design methodology (detailed comparisons can be found in Sect. 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 Sect. 3.2). To maximize leakage saving while minimizing the impact on performance, in this section, we discuss how the iTLB can be switched between the low-leakage mode the active mode at the appropriate time and in the appropriate manner.

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

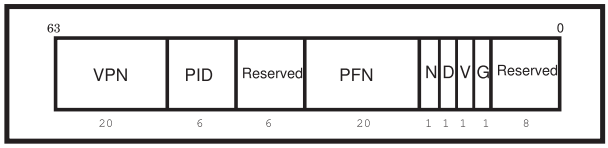
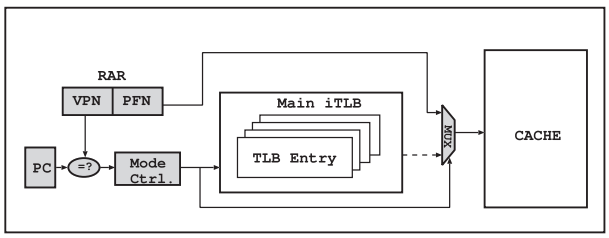
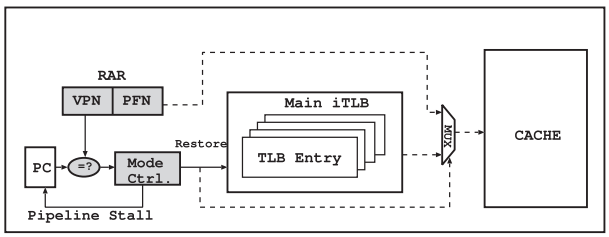


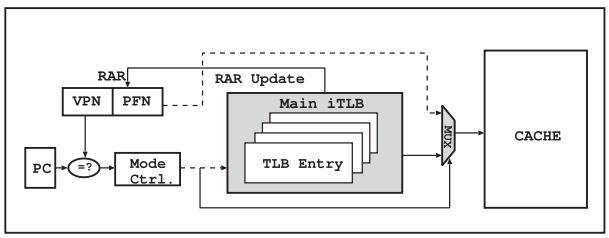
Fig. 3 TLB entry structure.



(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

Fig. 4 Working process of the 1-RAR policy.

employed as the higher hierarchy to filter out unnecessary main iTLB accesses [†]. As shown in Fig. 3, 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.

Figure 4 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).

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 paper. 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

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

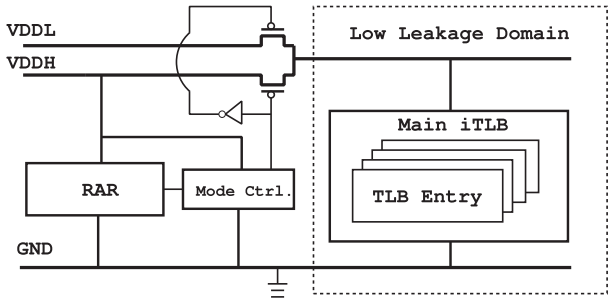


Fig. 5 iTLB design with the DVS technique.

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 paper, 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 inform iTLB whether the current reference is a WALKING one.

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 paper, 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 [2]. 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.

As shown in Fig. 5, a dual supply network is employed to provide fast switching between different supply voltages.

Table 3 Power parameters(@25°C).

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

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 32λ [†] 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 200 MHz target frequency. In the following sections, the mode transition penalty will be designated as one clock cycle.

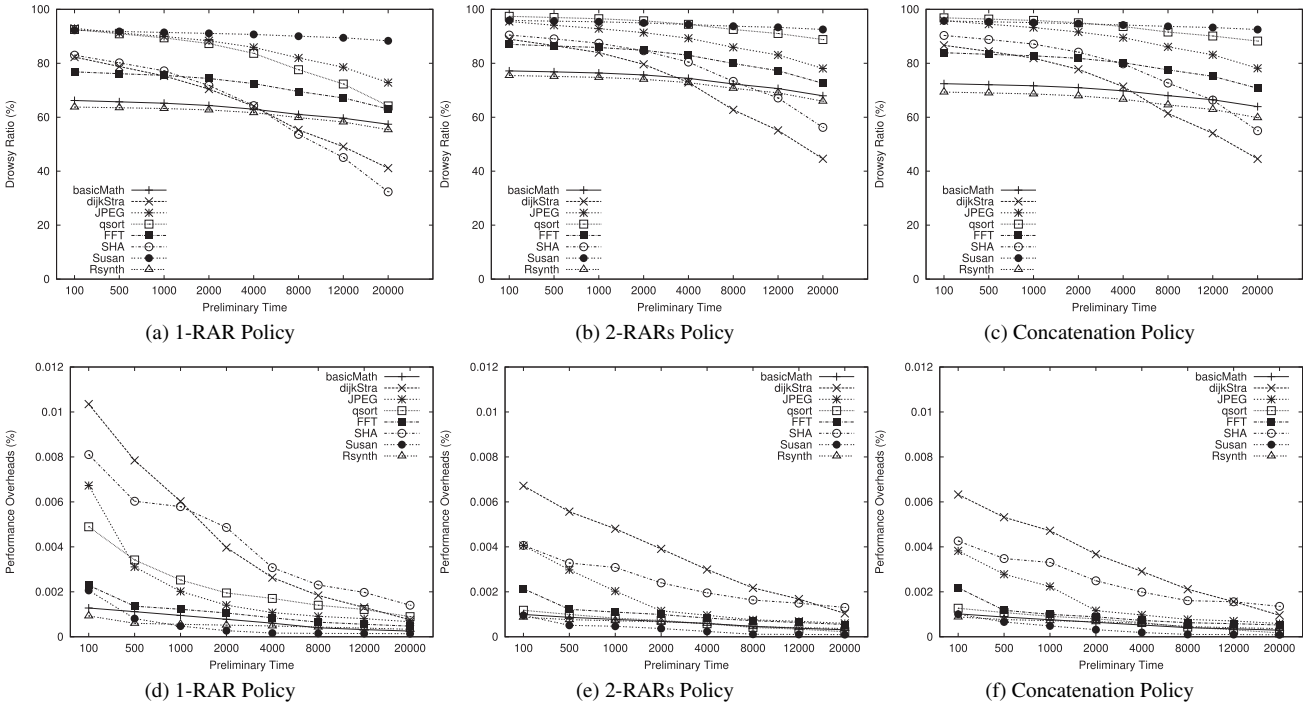
The soft error rate [12] is also a concern in very deep sub-micrometer technology. Since the soft error rate increases as supply voltage scaling, to alleviate the side effect caused by lowering voltage, a rather conserved voltage: 0.8 v, is selected as the lower voltage supply for the main iTLB.

Table 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 65 nm CMOS technology (202SZ) by Synopsys EDA tools (Design Compiler and Astro) [13]. The implemented iTLB obeys the specification of the MIPS R3000 processor [14], 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 [††], and treat the power consumption of such a fraction as the average power of the whole application programs. In Table 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 result by HSIM [13].

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 paper, all evaluations is based on the experimental infrastructure presented in Sect. 2, and the base-line

[†] λ equals to the half of the minimum transistor channel length
[††] As shown in Fig. 1, the one-entry miss rate of 'SHA' is in the 4th place of all 8 applications.



(a) 1-RAR Policy (b) 2-RARs Policy (c) Concatenation Policy

(d) 1-RAR Policy (e) 2-RARs Policy (f) Concatenation Policy

Fig. 6 iTLB drowsy ratio & performance overheads comparison.

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 Sect. 4.1, an approximate dynamic power reduction result is also presented.

4.1 Basic Evaluation

Figure 6 (a) ∼ Fig. 6 (f) 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 paper, 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 Sect. 3. The power evaluation model can be expressed as following:

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

$$D_{new} = D_{filter} + (1 - P_{Drowsy}) \times D_{iTLB} + D_{counter} + D_{transition}, \quad (2)$$

In Eq. (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 Eq (2), D_{new}, D_{filter}, D_{iTLB} and $D_{counter}$ are the dynamic power counterpart of the Eq. (1), while the $D_{transition}$ is the dynamic power consumed by drowsy-to-active mode transitions.

Figure 7 and Fig. 8 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 3.4 μW and 0.308 μW, 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:

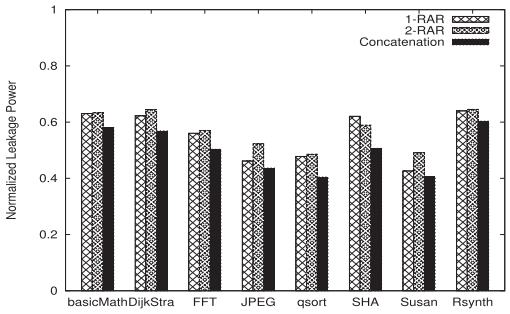


Fig. 7 Normalized leakage power consumption.

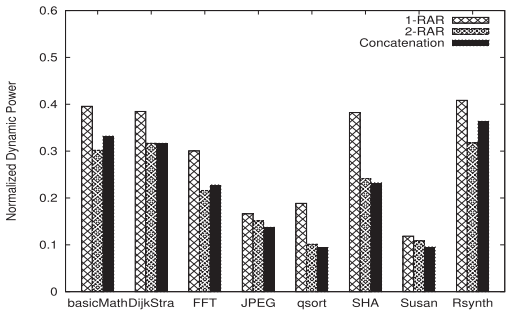


Fig. 8 Normalized dynamic power consumption.

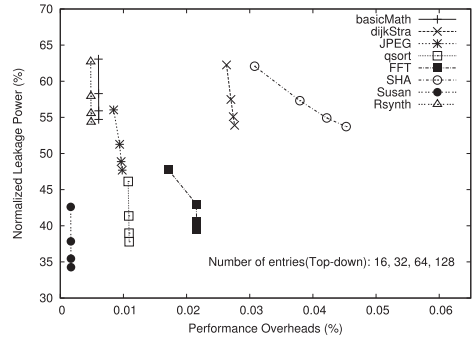
$$D_{transition} = \frac{N_{transition}}{\frac{clock\_cycles}{program}} \times \frac{E_{transition}}{\frac{seconds}{clock\_cycle}}, \quad (3)$$

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 Eq (3) equals to the performance overheads, which, as shown in Fig. 6 (d) ∼ Fig. 6 (f), are less than 0.01%. Therefore, the mode transitions with proposed policies only have a negligible contribution to the total dynamic power.

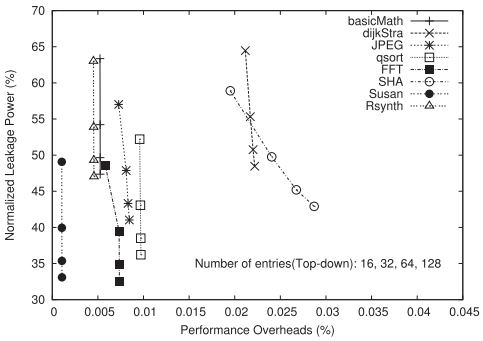
As shown in Fig. 7 and Fig. 8, while the Concatenation policy has the best leakage reduction effect, dynamic power saving results are highly dependent on the application programs, and the 2-RARs policy slightly outperforms the Concatenation policy. If the spatial locality of an application program is 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 64 bits 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%.

4.2 Design Scalability

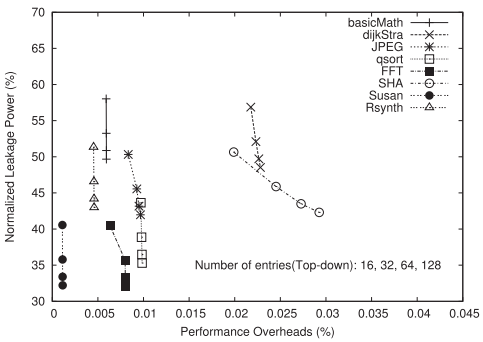
All above evaluation results are based on a 16-entry baseline configuration. To verify the scalability of the proposed design, additional evaluations are also executed by varying the size of iTLB. Figure 9 (a) ∼ Fig. 9 (c) show the robustness testing results with 3 different policies, where the horizontal 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) 1-RAR Policy



(b) 2-RARs Policy



(c) Concatenation Policy

Fig. 9 Leakage reduction efficiency with varying a iTLB size.

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 the extra leakage power of the second RAR becomes less significant for large-sized iTLBs. As shown in Fig. 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. Related Work

Previous publications on low-power TLB design are mainly focused on dynamic power. One of the low-power TLB structure is the block buffering [5], [15], 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.

The banked TLB [6] 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 [16], [17] 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.

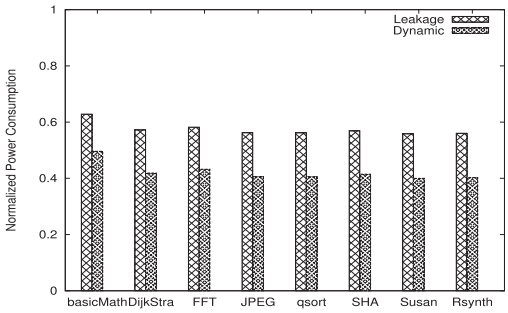


Fig. 10 Normalized power consumption with drowsy cache structure: Leakage & dynamic.

A possible alternative scheme for leakage-efficient iTLB design is to use the low-leakage mechanism of the drowsy cache [2]. 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. Figure 10 presents the power reduction effects of a 16-entry iTLB which simply adopts 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. 10, leakage power of the iTLB is reduced by 43% on average, and dynamic power is saved by 57%. Comparing with Fig. 7 and Fig. 8, 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 [11] 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 paper, 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.

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 paper). 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 Sect. 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. 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. When being applied to a MIPS R3000 processor (5-stage pipeline, 8 K 2-way instruction and data caches), the Dual-Vth design can reduce the leakage energy of the iTLB by 42%, while ours can reduce as much as 50% [†]. As a result, our design reduces 8% more leakage energy than the Dual-Vth design and incurs less performance degradation.

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 that 50% of the leakage power and 75% of the dynamic power can be reduced, at a cost of 0.01% performance degradation.

[†]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.

Acknowledgments

The authors would like to thank VLSI Design and Education Center (VDEC), Synopsys, Cadence, STARC, and Japan Science and Technology Agency (JST) CREST for their support.

References

[1] S. Kaxiras, Z. Hu, and M. Martonosi, "Cache decay: Exploiting generational behavior to reduce cache leakage power," Proc. 28th Annual International Symposium on Computer Architecture, pp.240–251, 2001.
[2] N. Kim, K. Flautner, D. Blaauw, and T. Mudge, "Circuit and microarchitectural techniques for reducing cache leakage power," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.12, no.2, pp.167–184, 2004.
[3] Z. Hu, A. Buyuktosunoglu, V. Srinivasan, V. Zyuban, H. Jacobson, and P. Bose, "Microarchitectural techniques for power gating of execution units," Proc. International Symposium on Low Power Electronics and Design, pp.32–37, 2004.
[4] N. Seki, Z. Lei, J. Kei, D. Ikebuchi, Y. Kojima, Y. Hasegawa, H. Amano, T. Kashima, S. Takeda, T. Shirai, 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. IEEE International Conference on Computer Design 2008, vol.182, 2008.
[5] L. Clark, B. Choi, and M. Wilkerson, "Reducing translation lookaside buffer active power," Proc. 2003 International Symposium on Low power Electronics and Design, pp.10–13, 2003.
[6] Y. Chang, "An ultra low-power TLB design," Proc. Conference on Design, Automation and Test in Europe: Proc., pp.1122–1127, 2006.
[7] 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., vol.10, no.2, pp.229–257, 2005.
[8] QEMU, "http://fabrice.bellard.free.fr/qemu"
[9] M.R. Guthaus and J.S. Ringenberg, "Mibench: A free, commercially representative embedded benchmark suite," 2001 IEEE International Workshop on Workload Characterization, WWC-4, pp.3–14, Dec. 2001.
[10] K. Matsuo, M. Sato, and M. Namiki, "Development of system evaluation environment using QEMU for low power system," Proc. Symposium on Advanced Computing Systems and Infrastructures, pp.61–68, 2007.
[11] R. Giorgi and P. Bennati, "Reducing leakage in power-saving capable caches for embedded systems by using a filter cache," Proc. 2007 Workshop on Memory Performance: Dealing with Applications, Systems and Architecture, pp.97–104, 2007.
[12] M. Zhang and N. Shanbhag, "Soft-error-rate-analysis (SERA) methodology," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol.25, no.10, pp.2140–2155, 2006.
[13] SYNOPSYS, "http://www.synopsys.com"
[14] D. Sweetman, See MIPS Run, Morgan Kaufmann, 2006.
[15] D. Fan, Z. Tang, H. Huang, and G. Gao, "An energy efficient TLB design methodology," Proc. 2005 International Symposium on Low Power Electronics and Design, ISLPED'05, pp.351–356, 2005.
[16] Y. Chang and M. Lan, "Two new techniques integrated for energy-efficient TLB design," IEEE Trans. Very Large Scale Integr. (VLSI)

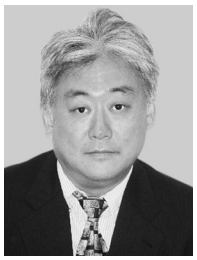
Syst., vol.15, no.1, pp.13–23, 2007.
[17] J. Lee, G. Park, S. Park, and S. Kim, "A selective filter-bank TLB system," Proc. 2003 International Symposium on Low Power Electronics and Design, pp.312–317, 2003.

Zhao Lei is currently a Ph.D. candidate at Keio University. His research interests include Microprocessor Achitecture and Low Power Design.

Hui Xu received the B.A. from Wuhuan University, China in 2002, and M. Edegrees from Keio University, Japan in 2008. Her research interests include the area of multi-core and low power processor design.

Daisuke Ikebuchi Currently a master course student in the graduate school of Sience and Technology, Keio University.

Tetsuya Sunata is a master's course student in the graduate school of Engineering, Tokyo University of Agriculture and Technology. He earned his Bachelor Engineering degree from Tokyo University of Agriculture and Technology in 2008.



Mitaro Namiki is a professor in the department of Computer Science, faculty of Technology, Tokyo University of Agriculture and Technology. He received Ph.D. from Tokyo University of Agriculture and Technology in 1992. His research interests include Operating Systems, Programming Languages, Parallel Processing, Computer Network.

Hideharu Amano received the Ph.D. degree from the Department of Electronic Engineering from Keio University in Yokohama, Japan in 1986. He is currently a professor in the Department of Information and Computer Science, Keio University. His research interests include parallel architectures and reconfigurable systems.