PAPER

# A Leakage Efficient Data 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 data TLB (Translation Look-aside Buffer) design for embedded processors. Due to the data locality 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 (with lower voltage supply) 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%.

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

## 1. Introduction

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 represented the predominant factor in CMOS circuits for many years, leakage power, which is consumed by each transistor even when no active switching is taking place, is increasingly prominent with technology scaling. 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, 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 of the function unit) 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 normal mode. However, to guarantee the overall leakage reduction effect, leakage reduction mechanisms on other components are also necessary.

Manuscript received April 21, 2010.
Manuscript revised August 23, 2010.

[†]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.51

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 high 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 the 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 "hotspots", due to their high power density. For instance, according to the simulation results from paper [7], the power density of a data TLB is 3 times higher than that of a data cache. Since leakage power varies exponentially with the temperature, TLBs are usually one of the most "leaky" components on a processor.

This paper focuses on reducing the leakage power of the data TLB (dTLB)**. Although TLBs have a cache-like structure, blindly transplanting cache leakage reduction mechanisms, such as cache decay [1] and drowsy cache [2], 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 ref-

*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 instruction TLB has a very different referencing pattern, its leakage reduction mechanism will be another work.

erences in and between adjacent slices can be utilized and contributive dTLB entries[†] 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.

The remainder of this paper is organized as follows. The next section presents the experimental platform for this work. In Sect. 3, the basic design philosophy and detailed leakage reduction mechanism will be illustrated. Leakage saving results will be shown in Sect. 4, and we will discuss related work in Sect. 5. Section 6 concludes the paper.

## 2. Experimental Setup

Before going to the detailed leakage reduction mechanism, we first show the experimental environment on which the analysis and evaluation in this paper are based. In our design, the MIPS system emulator from QEMU [8] 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 [9] 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 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 [10]. The number of TLB-flushes of each application program is shown in Table 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.

The power evaluation in this paper is based on the post-layout simulation. We implemented a 16-entry dTLB with Fujitsu 65 nm CMOS technology using Synopsys EDA tools (Design Compiler and Astro) [11]. The implemented dTLB obeys the specification of the MIPS R3000 processor [12], 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 paper is obtained with Synopsys' PowerCompiler, while the leakage reduction ratio achieved by DVS and mode transition overheads come from the circuit simulation with HSIM [11].

**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 |

## 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.

### 3.1 dTLB Referencing Pattern Analysis

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. Figure 1 shows the distribution of the temporary footprint of eight application programs at

[†]In this paper, 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.

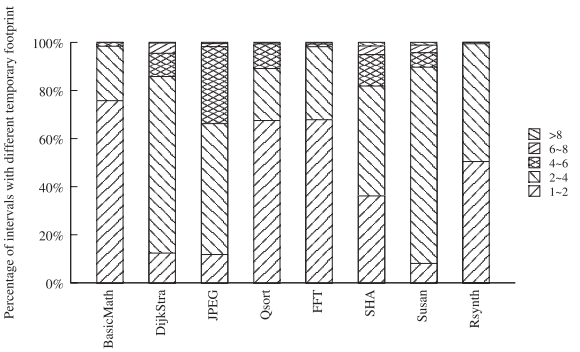



**Fig. 1** Temporary footprint distribution.

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%.

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.

### 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[†] only when being accessed again. Such a mechanism is similar to that used in drowsy cache [2], 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. Figure 2 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.

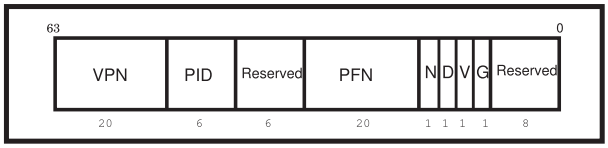



**Fig. 2** TLB entry structure.

As shown in Fig. 2, 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 Sect. 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. Figure 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

[†]Indicating voltage supply is restored to the higher voltage.

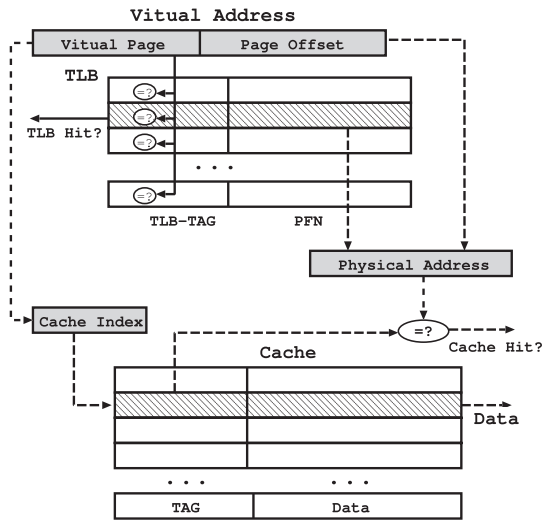



**Fig. 3** TLB-TAG path.

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 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.

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.

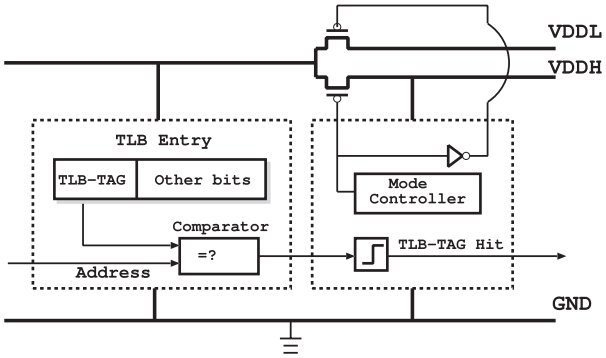



**Fig. 4** dTLB entry schematic.

**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.

### 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 [2]. 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.

As shown in Fig. 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 $16\lambda^{\dagger}$ header PMOS with each-entry control granularity, the mode transition time for a dTLB entry is about 1.9 ns, which is less than one clock cycle for our 200 MHz target frequency. In the following subsections, the mode transition penalty will

$^{\dagger}\lambda$ equals to the half of the minimum transistor channel length.

**Table 3** Power parameters.

| Leakage | |
|---|---|
| Active 16-entry | 37.8 $\mu$W |
| Active 1-entry | 3.3 $\mu$W |
| Drowsy 1-entry (0.9v) | 1.4 $\mu$W |
| **Dynamic** | |
| 16-entry | 688.6 $\mu$W |
| 1-entry | 14.1 $\mu$W |

be designated as one clock cycle.

Figure 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. Equation (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 $F_{max}$.

$$V_{norm} = V_{th}/V_{max} + (1 - V_{th}/V_{max}) * F_{norm}, \quad (1)$$

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 3 lists the power parameters of the proposed design, which are obtained from the post-layout simulation as mentioned in Sect. 2.

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

### 4.1 Basic evaluation

In Sect. 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.

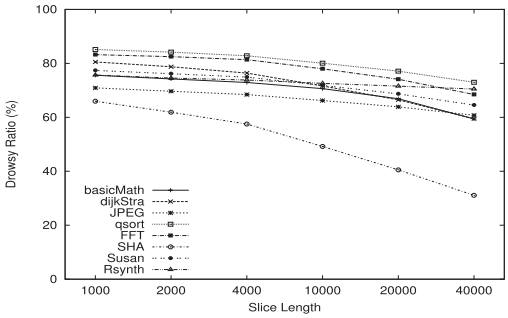



**Fig. 5** dTLB drowsy ratio: Slice length.

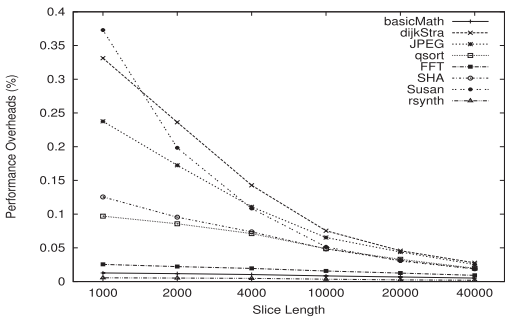



**Fig. 6** dTLB performance overheads: Slice length.

Figure 5 and Fig. 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 7 and Fig. 8 show the drowsy ratio and performance overheads under different (history length (HL), fast-wake-up threshold (FT)) configurations. In Fig. 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. 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

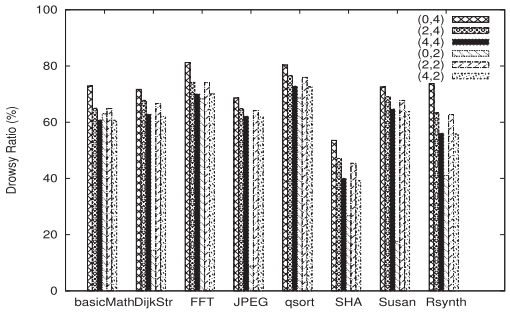



**Fig. 7** dTLB drowsy ratio: HL & FT.

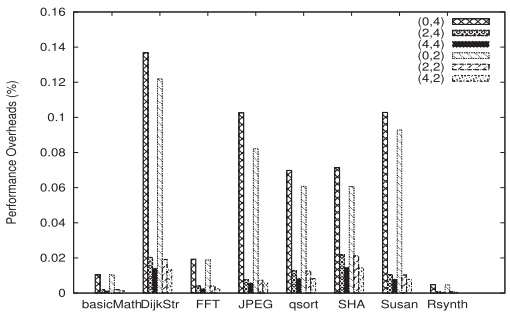



**Fig. 8** dTLB performance overheads: HL & FT.

the worst performance overheads is less 0.3%, is selected for the purpose of power evaluation.

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

$$L_{new} = (1 - P_{Drowsy}) \times L_{entry} \times N_{entry} + L_{counter} + L_{trace} + P_{Drowsy} \times L_{entry_L} \times N_{entry}, \quad (2)$$

$$D_{new} = (1 - P_{Drowsy} + P_{Drowsy\_acc}) \times D_{entry} \times N_{entry} + D_{counter} + D_{trace} + D_{transition}, \quad (3)$$

In Eq. (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. 7. As mentioned earlier, the $L_{entry_L}$ is measured by 0.9v voltage supply.

In Eq. (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. 8.

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

**Table 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% |

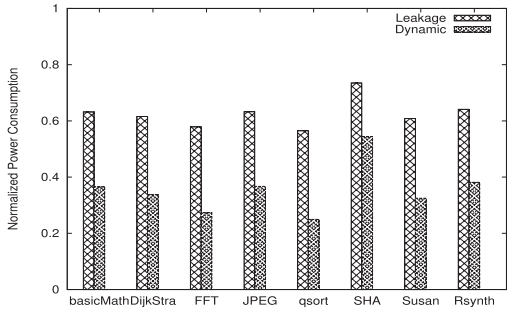



**Fig. 9** Normalized power consumption: Leakage & dynamic.

$$D_{transition} = \frac{N_{transition}}{T_{exec}} * E_{transition}, \quad (4)$$

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 $6.23 \times 10^{-14}$ 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 4 shows transition ratio which equals to the first part of Eq. (4) in the percentage form.

Figure 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-clock-cycle-counter is 3.4 $\mu$W and 0.308 $\mu$W, and the trace register consume 3.6 $\mu$W dynamic power and 0.4 $\mu$W 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%.

### 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.

Figure 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 pre-

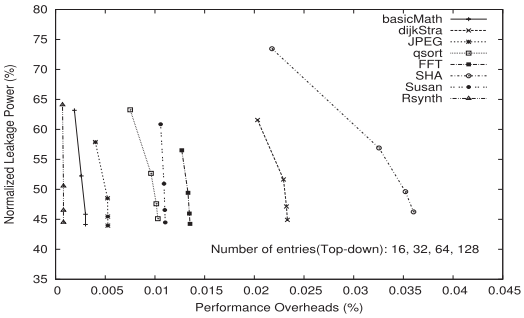



**Fig. 10** Leakage reduction efficiency with varying size dTLBs.

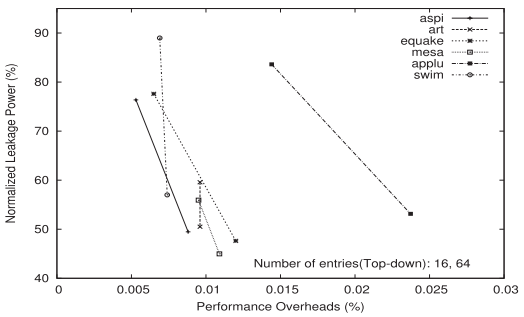



**Fig. 11** Leakage reduction efficiency on data intensive applications.

sented 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. 10 – as the dTLB size increases, more significant 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.

Figure 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 [13]. 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.

## 5. Related Work

Previous researches on low-power TLB design are mainly focused on dynamic power. One of the low-power TLB structure is the block buffering [5], [14], 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 [7] 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.

The banked TLB [6] 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 [15], [16] 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.

Another research which is closely related with ours is the drowsy cache [2]. 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 32 KB L1 cache by 52% on average. However, unlike the

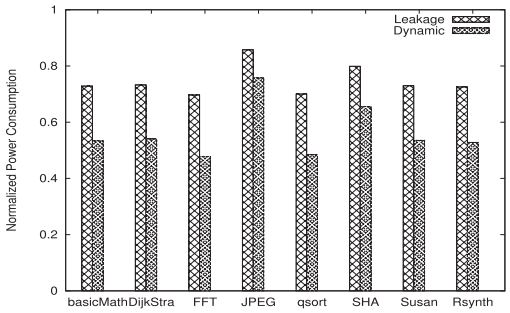



**Fig. 12** Normalized power consumption with DS mechanism: Leakage & dynamic.

cache design, the TLB-TAGs occupy a signification portion of the whole TLB. As mentioned in Sect. 3, if TLB-TAGs of all entries are 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. Figure 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 Sect. 3. The design parameters for power evaluation are the same as mentioned in Sect. 4, and we assume the activation of TLB-TAGs consumes no extra power. As shown in the Fig. 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. 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. 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.

## Acknowledgements

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

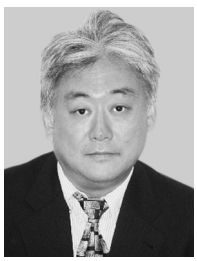
[1] S. Kaxiras, Z. Hu, and M. Martonosi, "Cache decay: Exploiting generational behavior to reduce cache leakage power," Proc. 28th annual international symposium on Computer architecture, pp.240–251, 2001.
[2] N. Kim, K. Flautner, D. Blaauw, and T. Mudge, "Circuit and microarchitectural techniques for 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, 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.S.K. Matsuo 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.
[10] M.R. Guthaus and J.S. Ringenberg, "Mibench: A free, commercially representative embedded benchmark suite," 2001 IEEE International Workshop on Workload Characterization, 2001. WWC-4, pp.3–14, Dec. 2001.
[11] SYNOPSYS, "http://www.synopsys.com"
[12] D. Sweetman, See MIPS Run, Morgan Kaufmann, 2006.
[13] SPEC2000, "http://www.spec.org"
[14] 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, 2005. ISLPED'05, pp.351–356, 2005.
[15] 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.
[16] J.H. Lee, G.H. Park, S.B. Park, and S.D. 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.E. degrees from Keio University, Japan in 2008. Her research interests include the area of multi-core and low power processor design.

**Daisuke Ikebuchi** is currently a master course student 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.