社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2008－ARC－179 (21)
2008／8／7

Reducing Power of TLB with Power-Gating Technique on Microprocessor

Xu Hui,† Lei Zhao,† Tetsuya Sunata,†† Naomi Seki,†
Mitaro Namiki†† and Hideharu Amano†

The TLB(translation lookaside buffer) is the hardware that translates the virtual address used by program to the physical address, which accesses memory. This high associate structure consumes considerable power of the microprocessor about 16%. The leakage of register file is also a problem, while prior work only has looked into reducing dynamic power of TLB. In this paper, we use fine-grained RTPG(run-time power gating) technique to reducing TLB leakage power. The main idea is: for iTLB, using a recently accessing register to save the sequential accessed entry, then power-off the whole iTLB file; for dTLB using counters on every entry, when counters exceeds a threshold power-off that entry line. Results with a suite of Mibench mark shows that with the methods, for iTLB 67% leakage power can be saved, for dTLB 41% leakage power can be saved. Despite the small increase of miss rates, the approach can reduce leakage power of TLBs, without damaging the performance of microprocessors.

1. Introduction

In recent advanced portable machines, as the complex calculation units and multiple functions are needed, the power consumption becomes the crucial design issue. Low power consumption for conserving battery energy on embedded devices while high performance takes an inevitable consequence on chip design. Reducing the power dissipation requires an in-depth examination of each system component, and over the past few years researchers have become very active in this area. Power consumption is classified into two types; the dynamic power needed when the elements are accessed or activated, and the leakage power consumed even when the elements are idle. The leakage power becomes more important in future advanced processor, however today's main concern is still focused on reducing dynamic power except the main part of CPU.

TLB is on the critical path of memory system look up, in order to reduce the miss ratio, it uses high degree of associativity which consumes considerable power. In fact for some commercial processors like Strong ARM and Hitachi SH-3, the TLB consumes about 16% of the total power of the chip[1]. As In our project named Geyser, in the first version the TLB occupied the 36% of the total power compared to the CPU core. That recently draws researchers' attention to TLB power consumption. However, most of the related work focuses on reducing dynamic power of TLB, in this paper we focus on reducing the leakage power of TLB by using fine-grained Run Time Power-Gating technique. Power-Gating (PG) is a technique for efficiently reducing leakage power by shutting off the idle blocks.

1.1 Runtime Power-Gating

The traditional PG is suitable for large semiconductor domain, like entire CPU or entire IP. The High Vth power switches (sleep transistors) are inserted between Ground(GND) and Virtual Ground(VGND), by turning off the sleep transistors, the power for the entire domain is shut off. That is called the sleep mode. The time for switching from the sleep mode to the normal mode is called wake-up time. But the wake-up time of traditional PG is long, usually an order of microseconds.

1.1.1 A Fine-Grained Runtime Power-Gating

As the locally-Shared Virtual ground(LVS) scheme is introduced, the target domain is partitioned into smaller local power domains. The VGND line and power switch are shared only within the local power domain. Power of multiple local power domains are controlled by a single sleep control signal. With this method, the wake-up time is fast, usually a few nanoseconds. Of course PG cells and a different design method are necessary. The fine-grained RTPG technique is considered to be applied on TLB.

1.1.2 Break Even Time

PG technology is not zero-penalty. Energy dissipation at a sleep event is the sum of dynamic energy consumed at the sleep-in and wake-up operations and leakage energy. Power-off only gets gain in energy savings when sleep time exceeds break-even point. It's important to analyze the BET to decide the shut-down and wake-up event on the PG unit.

† Graduate School of Science and Technology, Keio University
†† Department of Computer and Information Sciences,Tokyo University of Agriculture and Technology

1.2 An Example-Geyser0

Geyser is a project which dives into developing very low power microprocessor especially for embedded applications. A prototype chip Geyser-0, MIPS3000 compatible processor, is designed and implemented with 90nm CMOS technology. In this version, a fine grained dynamic PG dynamically controls shut-off and wake-up individual computational units in the CPU according to the current executing instructions. A normal MIPS R3000 compatible core and a power gated counterpart are both implemented on a single chip, while a single bridge unit which includes a 16-entry TLB unit, a Memory Management Unit (MMU), a 64Bytes x 64 Instruction Cache and a 64Byte x 64 Data Cache is used to implement the memory access requests of both the normal core and power-gated core. At present, for the consideration of area and power, a 64Bits x 16-Entry TLB is implemented. The instruction TLB and data TLB are combined, in other words instruction TLB and data TLB share the 16 entries, two ports in and two ports out. In the current design, because TLB in on the critical path of CPU, though the virtual address comes at negative edge of clock cycles we start the comparison on the next clock cycle. From the design result, by comparing to the normal CPU core Figure 1, TLB consumes 36% of the total power including 29% of the leakage power and 30% of the area.

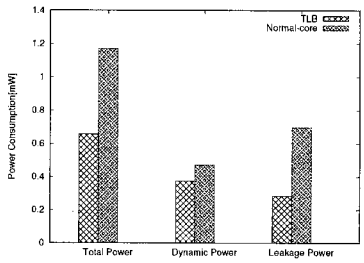

Fig. 1 Memory Access in Geyser0

2. Related Work

Power reduction techniques for TLB can be classified into three types, clock gating technique, banking technique and compiler optimization. Some paper[2] combined some of them together. However, all of them focus on reducing dynamic power by reducing access the whole entries of TLB.

2.1 Clock Gating with TLB

TLB accesses exhibit temporal and spatial locality, sequential memory accesses are often to the same page. A keep-register is provided to keep the page[3]. If the page accessed in the lookup cycle N is the same page as which is accessed in the lookup in cycle $N-1$, then the gate for clock is turned off for subsequent actions.

2.2 Banking Technique with TLB

In order to reduce the number of TLB accesses, the banked structure is a natural idea. It partitions the main TLB into several banks. By accessing only one bank can effectively reduce the power consumption per TLB access.[4]

2.3 Compiler-Directed Technique with TLB

Another low power TLB design is based on a compiler support. The main idea is to keep translations in a set of translation registers (TRs), and intelligently uses them in software to directly generate the physical addresses without going through the data TLB. By avoiding going through the dTLB (data TLB) whenever possible, in the paper[1], they save on the dynamic power consumption on those accesses. Instead of accessing a fully associative array, they select to use one of TRs.

3. Low Power TLB Methodology

The TLB, a small on-chip cache, is costly in terms of power consumption because its fully associative structure for translating virtual to physical address. In this section, we will propose two methods for instruction TLB(iTLB) and data TLB(dTLB) respectively based on the fine-grained RTPG technique. For iTLB, we used a RAR (recently accessed register) to save the recently hit entry information, and then the whole iTLB register can be shut down. For dTLB we use counters to calculate the elapsing of time without being accessed, if an entry counter exceeds a threshold time the entry will be shut down.

Table 1 Research On TLB Access

Bench	qsort	jpeg	dijStra	sha	FFT
itlb miss	0.0002%	0.0004%	0.0013%	0.0001%	0.0013%
itlb seq	91.62%	98.17%	83.38%	84.57%	76.86%
dtlb miss	0.180%	0.040%	0.104%	0.488%	0.606%
dtlb seq	56.29%	78.79%	65.18%	79.09%	77.44%

3.1 TLB accessing character analyzing

Here, we executed five applications from Mibench on MIPS4000 based virtual machine with embedded Linux. Table 1 shows the program sequential page access rates and miss rates for iTLB and dTLB. For instruction TLB, the sequential page access rate is high. However, for data TLB, the sequential page access rate is relatively low, especially when programs run with the operating system (OS). In order to apply different methods on TLB we separate TLB into iTLB and dTLB.

3.2 Methodology for iTLB

Based on the research on the accessing character of iTLB, the sequential memory access of iTLB takes

up more than 90% of the total access requests. That means the instructions of program are mostly fetched in the same physical page. Figure 2 shows the design of iTLB.

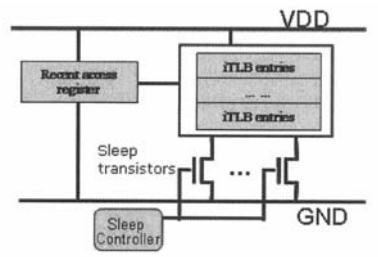

Fig. 2 iTLB Design Figure

When the program enters a physical page, we save the page number into a register called RAR(recently accessing register). If instructions are kept to be fetched from the same page, after a certain time (threshold time), the power for all of the iTLB registers is shut off. Note that every data in iTLB register is lost with the current fine-grained PG technique, since the power of flip-flops is completely removed. It is called Sleep-in Event, and only let the RAR at power-on mode be available. When the access goes beyond the page boundary or a branch instruction to other page is taken, the TLB miss happens, which causes the exception to refill iTLB entry. These operations are managed by OS in kernel mapped page which do not use TLB. Thus, during these operations iTLB registers can wake-up, and it is called Wake-up Event, by making of the use of quick wake-up of fine-grained RTPG. We can let the iTLB registers in the wake mode before the OS writes the new entry into the iTLB. When OS finished the exception routine, the new entry is in the iTLB without overhead of wake-up. Note that this method can also save the dynamic power by accessing the RAR first, then search the whole iTLB entries. When RAR hits, the power for searching registers can be saved even if they are not in the sleep mode.

On the other hand, sleep-in and wake-up operations will cause extra dynamic power. And also the content which saved inside the iTLB will lost during the sleep mode which will cause extra iTLB misses. So the threshold time to power-off the iTLB registers must be selected based on the evaluation results. And, we also keep the iTLB sleep enough long time which can exceed the Break Even Time to get power consumption gain.

3.2.1 The Design of RAR

The RAR(recently accessing register) acts the important role in the proposed method of iTLB. RAR has the same structure as one-entry TLB holding 64 bits for virtual page bits, physical page bits, flag bits and preserved 14 bits. So we can use these 14 bits in RAR to act as a counter to calculate the sequential access time in order to control the Power-Gating operation on iTLB. At the same time, RAR is a common technique among embedded processor to save dynamic power[3], naturally it can gain the same merit here. However, the value updating on RAR should be well cared to suppress the miss rate. If OS manages iTLB all cases, the RAR can be set by OS. In this case, if the miss is caused by branch, the OS can set the target page number to RAR in order to avoid the miss on the next access.

However, here we assume that the RAR is set by the hardware automatically. Update sequence of RAR is shown in Figure3. Here let the page number before miss be PGa and after miss be PGb. (1) When the RAR suffer a miss, iTLB is woken-up. (2) Refill exception caused by OS will update an entry to iTLB. (3) Copy the number of page to the RAR from one of the iTLB entry. (4) Update the RAR to PGb.

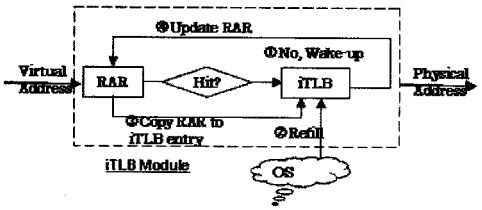

Fig. 3 RAR Update Rule

3.3 Methodology of dTLB

The following scheme is based on the assumption of random TLB entry replacement rule by OS which is mostly used in data TLB, although the other replacement schemes can also earn merit from the method. We aim to power-off unused entries to save power by fine-grained RTPG control on individual dTLB entry line shown in Figure 4. The judgment of unused entries can be done by using Counter or PID (process ID). Some entries which stored inside dTLB are not accessed long time, and it means that the access probability of such entry is quite low. Therefore, the leakage power dissipated on behalf of these entries might be wasted. We also can use the preserved 14 bits inside per entry as a counter, when the counter time exceeds a certain time (threshold time), then power of these entries are shut-off. If miss happened, the traditional random refill will caused by OS, if the refill target entry is in sleep mode, then OS wakes up the entry first then fill the content. The merit of this method

is that the miss rate is less than the method of iTLB, if we can set the adequate threshold time. When the sleep time exceeds the BET, we can gain power saving from the PG. On the other hand, every entry needs a PG cell will cause more area overhead and also more dynamic power. The maintenance of entry-number counters will also causes dynamic power. This scheme also can be used by iTLB, but the refill in the sleep entry will take time penalty. As we know that TLB is on the critical path of memory access, we selected the method using the RAR for iTLB. As an alternative method, we can control the power of each entry by using PID. That is, if the current process frequently accesses the entries with a PID, the others will not be accessed unless the OS switches its process. So, we can let the dTLB entries of current running process awake, leave the others in the sleep mode. Although this scheme is better than the first one, we need the cooperation of OS to get the current PID information, and when the process switch event happens, it will be informed to the dTLB. We will discuss it in the future. This time we will focus on the method with the usage counter.

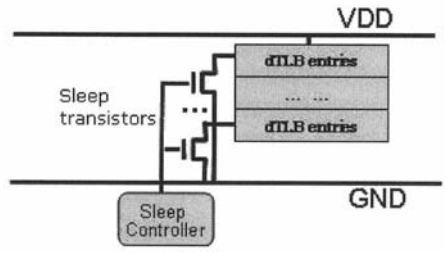

Fig. 4 dTLB Design Figure

4. Experimental Results

4.1 Experiment Setting

We used updated QEMU, a tool by a group[5] of authors for tracing memory access and executed instructions running on the virtual machine for x86, ARM and MIPS[6]. The parameters used in the evaluation are shown in Table 2. We obtained the data by executing five applications of different fields from MiBench, a free, embedded benchmark suite[7].

Table 2 Configured parameters.

Trace Environment	
CPU Type	Updated Debian MIPS
Instruction execution	In-order
OS Type	Linux 2.6.15
TLB	16-Entry Full-associate
Data/Inst. Page size	4KB

4.2 Evaluation of Threshold Time

We got the trace data by QEMU which includes the trace of virtual and physical address for fetched instructions, and those for data accesses. The application is running on the OS(operating system), so the address is not only the memory access of the application. It also includes the memory accesses such as exception handled by OS, the QEMU execution itself and others. The data is suitable for TLB analysis, since TLB is managed by OS rather than used by a single application.

These memory accesses data can be analyzed by simulating the TLB actions, and the Power-Gating iTLB and dTLB are also simulated with the methods described in the section 3. Here we make an assumption that the CPI (clock per instruction) is one, because the trace data does not include the required clock cycles. For evaluating miss rate and area consideration, we used an 8-Entry iTLB and 16-Entry dTLB.

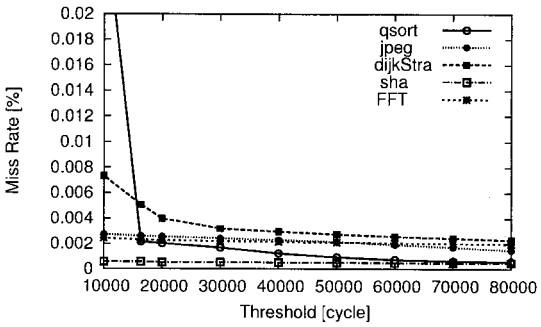

Fig. 5 iTLB Threshold Analysis

4.2.1 iTLB Threshold Time

The miss rate versus threshold time for iTLB shutdown is shown in Figure 5. The miss ratio is large when we set small number of threshold, because of the frequent miss of the RAR. From the figure we can find out that above 20000 cycles, when we shut the iTLB registers down, the variety of miss rate is small and the fluctuation is trivial as 10^{-4}. However, a 15-bit counter is needed for counting 2000 cycles, there is only 14 preserved bits inside the TLB entry. So, 16380 (2^{14}) data is also shown in the figure whose miss rate is comparable stable to satisfy the needs. Thus, for convenience sake we can set the threshold time to $16379(2^{14}-1)$ cycles when all the preserved bits becomes 1.

4.2.2 dTLB Threshold Time

Figure 6 shows the same graph as Figure 5 but for data TLB. Since the miss rate of dTLB is higher than

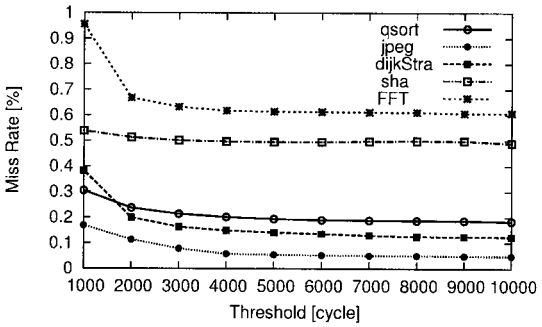

Fig. 6 dTLB Threshold Analysis

iTLB, we should be more careful to choose the threshold time. From the figure we find out that since 4000 cycles the miss rate of dTLB becomes nearly the same. The reason is that after 4000 cycles, if no accessing on an entry means that the probability for accessing that entry is very low. In LRU replacement policy, it may be replaced, that's why the miss rate becomes stable. For a 4000 corresponding 2^{12}, we needs 12-bit counter to control the shut-off power of entry line. For make the implementation simple, we can set the threshold to $4095(2^{12}-1)$ cycles when 12 bits become 1.

4.2.3 Threshold Time inference

The shutting threshold time for an 8-Entry iTLB is almost 5 times of shutting 1-Entry for dTLB. However, if we choose to shut the iTLB around dTLB threshold×8 that becomes between 30000 and 40000, and the miss rate will become more stable for iTLB. It infers that the power-off threshold time of N-Entry file is approximately N times of the threshold time of one entry number.

4.3 Evaluation of Power Consumption

4.3.1 Analytical Model

The power evaluation is difficult in terms of RTPG since it includes the transient power of shut-off, stable leakage power when the sleep transistor is turned off, and the power for wake-up. Here, we propose an analytical model to calculate the leakage saving of the methods described in the previous section. To simplify the whole evaluation process, we will ignore the overhead brought by counters, and assume the leakage power will be zero after entering the sleep-mode. Assuming the leakage energy of NPO(non-power-optimized) TLB is $E_{org} = P_{leak} \times t_{exesum}$, t_{exesum} is the program execution time, P_{leak} is the leakage when TLB is in wake mode.

For iTLB, the energy impact of PG events includes power-increasing parts:(a) the Energy E_{miss} of extra miss events caused by value loss in iTLB sleep mode, and (b) the Energy overhead $E_{iswitch}$ from power on/off the iTLB registers, and power-decreasing parts: (a) the Energy $E_{leak} = P_{leak} \times t_{sleep}$ saved by power off the iTLB register file and (b)the Energy E_{rar} saved by the fast accessing RAR as described in3.2.1. the energy saving can be represented with the following formula: $E_{save} =$

$$P_{leak} \times t_{sleep} + E_{rar} - E_{miss} - E_{iswitch}$$

The power consumption caused by miss event handling and mode transitions for each PG event can be treated as constant values; the difference of energy saving comes from the sleep time and sequential RAR access time. The over-all power saving of iTLB can be presented as: $E_{savesum} =$

$$(\sum_{i=1}^{n} P_{leak} \times t_{isleep}) + E_{rar} - E_{miss} - E_{iswitch}$$

where n is number of Power-gating times in the application.

For dTLB, when PG event occurs, the power consuming is represented similarly. But at each entry granularity the energy saving caused by a specified PG eventE_{save} is presented as follow:$E_{save}=$,

$$(\sum_{j=1}^{N} P_{leak} \times t_{jsleep}) - E_{miss} - (\sum_{j=1}^{N} E_{dswitch})$$

where N is the entry number, P_{leak} is the leakage power of a dTLB entry when in a wake-mode; the E_{miss} and the $E_{dswitch}$ are energy overheads caused by extra TLB misses and mode transitions; Assuming the n_{jsum} is the shut-off number of an entry-line for an application, the total saving energy is represented as follows:$E_{savesum} =$

$$= (\sum_{j=1}^{N} P_{leak} \times t_{jsleep}) - E_{miss} - (\sum_{j=1}^{N} E_{dswitch} \times n_{jsum})$$

en As stated in 1.1.2, the energy overhead caused by powering-on/off PG components equals to the leakage energy dissipated in a BET-period, which is $P_{leak} \times t_{bep} = E_{switch}$[8]. The TLB misses are treated as an exception, and it follows a standard handling process[9]. In our experiment data obtained by QEMU, when miss happens there always 19 instructions are called. So we can depict the extra leakage energy consumed by a TLB miss as: $E_{miss} = P_{leak} \times t_{miss}$ ignoring the extra dynamic power consumed by miss events. Assuming that M be the sum of miss increase times, let's predigest the formula to iTLB $E_{savesum} =$:

$$P_{leak} \times (\sum_{i=1}^{n} (t_{isleep} - t_{bep}) - t_{miss} \times M) + E_{rar}$$

,set N is the entry number, M_i is sum of miss increase times of 1-Entry dTLB $E_{savesum} =$:

$$P_{leak} \times (\sum_{j=1}^{N} (t_{jsleep} - n_{jsum} \times t_{bep}) - \sum_{i=1}^{N} M_i \times t_{miss})$$

4.3.2 The Evaluation Result of Power

By using the formulas above, without including the dynamic power saved by RAR, we can only consider the execution summary time, PG times, sleep time, BET time and TLB miss penalty time to calculate the ratio, Table 3 shows results. This time we use BET assumption based on the BET results of PG Units of Geyser1.2. Here BET is assumed to be 200 cycles for iTLB and 100 cycles for dTLB. "Power save" shows the leakage power saved by the PG, and "the effect PG" means the percentage of the sleep cycles exceed the BET. "Miss" is the original miss rate of benchmarks, while "PG miss" is the rate of using PG on iTLB. "Sleep time" shows the ratio of the sleeping in the total time.

Table 3 TLB Evaluation Result

iTLB Evaluation Result

Bench	qsort	jpeg	dijStra	sha	FFT
Sleep time	73.4%	83.6%	45.3%	79.0%	65.1%
Power Save	73.21%	83.39%	45.09%	78.89%	64.96%
Effect PG	99.55%	99.70%	98.74%	92.23%	96.76%
Miss	0.0002%	0.0004%	0.0013%	0.0001%	0.0013%
PG Miss	0.0022%	0.0261%	0.0050%	0.0001%	0.0023%

dTLB Evaluation Result

Bench	qsort	jpeg	dijStra	sha	FFT
Sleep time	45.8%	57.6%	38.1%	55.2%	11.8%
Power Save	45.31%	57.39%	37.73%	54.45%	10.94%
Effect PG	96.38%	98.25%	98.98%	92.23%	99.58%
Miss	0.180%	0.040%	0.104%	0.488%	0.606%
PG Miss	0.194%	0.054%	0.141%	0.495%	0.614%

Apparently the RTPG technique is a kind of technique that reduces leakage power but on the other hand it will increase miss rate. From the table, the iTLB can save average 69% leakage power of the total one. The dTLB can save average 41% leakage power of the total one. From the miss rate, we find that iTLB miss rate is average 10 times compare to the iTLB without using PG. For dTLB miss rate is average 1.09 times of the dTLB without using PG. In iTLB the miss rate is very small, even 10 times, that can hardly influence the performance of the CPU. In fact, the miss penalty is 19 CPIs in our experiments, and if we need low miss rates of iTLB we can adjust threshold to 40000 in which the miss rate is only 2 or 3 times of the traditional iTLB. But for dTLB the miss rate is inhere high which will influence the performance of CPU, and so low miss rate increase is important.

5. Conclusions

Different Power-Gating methodologies of iTLB and dTLB were proposed to save the leakage power based on the accessing characteristics and miss rates. By applying the schemes to the five applications from different fields of Mibench, we can see for iTLB 69% leakage power can be saved, besides of the dynamic power saving by the RAR, and for dTLB 41% leakage power can be saved. Though the RTPG will increase the miss rates, from our results the damage seldom influences the performance of processor. Besides the power saving, we also gave out the threshold time for power-off event. From the analysis of threshold time of iTLB and dTLB, we infer that the threshold time is depending on the register entry numbers. This time we do not get the precise time for BEP, in the future we will use the SPICE the analysis the BEP of registers. And also we should apply the low power design to real chip, and evaluate the power consumption based on our methodologies in the future.

6. Acknowledgments

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

References

1) Kadayif, I., Nath, P., Kandemir, M. and Sivasubramaniam, A.: Reducing Data TLB Power via Compiler-Directed Address Generation, *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, Vol. 26, No. 2, pp. 312–324 (2007).
2) Kadayif, I., Sivasubramaniam, A., Kandemir, M., Kandiraju, G. and Chen, G.: Optimizing instruction TLB energy using software and hardware techniques, *ACM Trans. Des. Autom. Electron. Syst.*, Vol. 10, No. 2, pp. 229–257 (2005).
3) Clark, L., Choi, B. and Wilkerson, M.: Reducing translation lookaside buffer active power, *Proceedings of the 2003 international symposium on Low power electronics and design*, pp. 10–13 (2003).
4) Chang, Y.: An ultra low-power TLB design, *Proceedings of the conference on Design, automation and test in Europe: Proceedings*, pp. 1122–1127 (2006).
5) Kazuya Matsuo, M. S. and Namiki, M.: Development of System Evaluation Environment using QEMU for Low Power System, pp. 61–68 (2007).
6) QEMU: http://fabrice.bellard.free.fr/qemu.
7) Guthaus, M., Ringenberg, J., Ernst, D., Austin, T., Mudge, T. and Brown, R.: MiBench: A free, commercially representative embedded benchmark suite, *Workload Characterization, 2001. WWC-4. 2001 IEEE International Workshop on*, pp. 3–14 (2001).
8) Jiang, H., Marek-Sadowska, M. and Nassif, S.: Benefits and Costs of Power-Gating Technique, *Computer Design, 2005. Proceedings. 2005 International Conference on*, pp. 559–566 (2005).
9) Sweetman, D.: *See MIPS Run*, Morgan Kaufmann (2006).