An Ultra Low-Power TLB Design

Yen-Jen Chang
Department of Computer Science, National ChungHsing University, Taiwan
ychang@cs.nchu.edu.tw

Abstract

*This paper presents an ultra low-power TLB design, which combines two techniques to minimize the power dissipated in TLB accesses. In our design, we first propose a real-time filter scheme to eliminate the redundant TLB accesses. Without delay penalty the proposed real-time filter can distinguish the redundant TLB access as soon as the virtual address is generated. The second technique is a banking-like structure, which aims to reduce the TLB power consumption in case of necessary accesses. We present two adaptive variants of the banked TLB. Compared to the conventional banked TLB, these two variants achieve better power efficiency without increasing the TLB miss ratio. The experimental results show that by filtering out all the redundant TLB accesses and then minimizing the power consumption per TLB access, our design can effectively improve the Energy*Delay product of the TLBs, especially for the data TLBs with poor spatial locality.*

1. Introduction

The *translation lookaside buffer* (TLB) is an essential component to speed up the virtual-to-physical address translation. Due to the high frequency of access, however, the power consumption of the TLB is usually considerable. Studies show that the power dissipated by the TLBs is usually a significant part of the total chip power. For example, in Hitachi SH-3 processor, the TLBs consume about *15%* of the total chip power [1], and the StrongARM processor (SA-110), which targets on low power applications, dissipates about *17%* of the total chip power in the TLB accesses [2].

The low power TLB designs can be broadly classified into two categories. One is to reduce the number of TLB accesses [1][3][4], and the other is to reduce the power consumption per TLB access [5][6]. Block buffering is a common technique [4]. If a reference hits in the block buffer, a translation is quickly returned without looking up the TLB. Otherwise, another cycle must be paid to access the main TLB. Kadayif et al. [1] proposed a scheme that combines the compiler with hardware enhancements to reduce the instruction TLB (iTLB) power. Their method is to keep the latest translation in a register and access the iTLB only in page change. The banked TLB [5] 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. Choi et al [6] proposed a two-level TLB architecture that integrates a 2-way banked filter TLB with a 2-way banked main TLB. By distributing the accesses to TLB entries across the banks in a balanced manner, their method optimizes the power consumption per TLB access.

In this paper, we first propose a real-time filter scheme to facilitate the block buffering to reduce the power consumption without delay penalty. The real-time filter introduces a new register design, referred to as *content-change-aware* (CCA) register. Unlike the traditional register which is insensitive to the stored value, a pronounced feature of the CCA register is that it can sense the content change in real-time. The key idea behind our design is to embed the XOR logic into the flip-flop. It would result in *47%* and *78%* increase in area cost and power consumption of register, respectively. Because the CCA register is only applied to the specific address registers, which occupy a tiny portion of chip area, these penalties are negligible to the entire chip.

The proposed second technique is to reduce the power consumption per TLB access in the case of block buffer miss, so the power dissipated in TLB accesses can be further reduced. Based on the banking technique, we present two variants of the banked TLB: VB1 and VB2. Unlike the conventional banked TLB whose disadvantage is the increased miss ratio, our design uses a different allocation to reduce the high miss ratio incurred by banking. This implies that our design can combine the high power efficiency of the conventional banked TLB and the low miss ratio of FA TLB.

We use *SimpleScalar* to perform the simulation of *SPEC2000*. All the power data are obtained from the *CACTI* tool and the *HSPICE* simulation of the extracted layout in TSMC 0.18μm technology with a 1.8V supply. The results conclude that due to the low miss ratio our design can improve the Energy*Delay product by about 41% compared to a fully-associative (FA) data TLB without banking, and 21% compared to a FA data TLB with conventional banking. Because the instruction TLB has very strong locality, the Energy*Delay product improvement is insignificant.

2. Real-Time Filter

Despite the power efficiency, the most serious drawback of the block buffering [4] is the lengthened access delay. This performance penalty can be hidden by overlapping the block buffer lookup with either the set decoding or tag comparison, but it is particularly hard to be hidden in the high performance

This work was supported by the National Science Council of Taiwan under grant no. NSC93-2213-E-005-027.

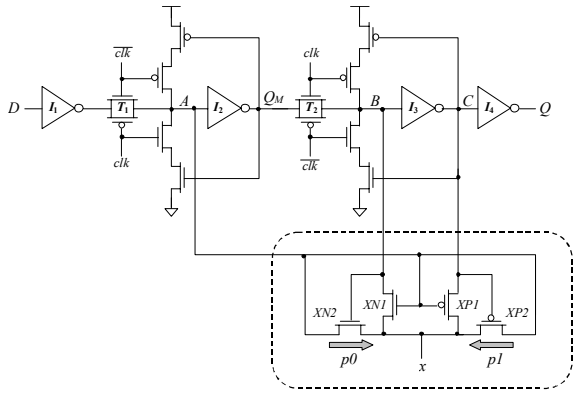

Fig. 1. The implementation of the CCA flip-flop.

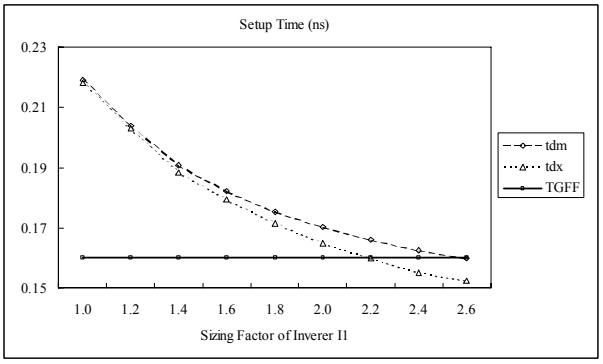

Fig. 2. The t_{dm} and t_{dx} delays of the CCAFF in different sizing factor of inverter I_1.

microprocessors. To avoid performance degradation and achieve absolutely accurate in filtering out the redundant TLB accesses, our real-time filter design aims to exactly identify the block buffer hit as soon as its virtual address is resolved.

2.1 CCA Register Design

A register, by definition, is a set of flip-flops in which a common clock is used for each flip-flop. To make a flip-flop aware of content change, we propose a variant flip-flop design, called *content-change-aware flip-flop (CCAFF)*. Because the transmission gate flip-flop (TGFF), derived from the PowerPC630, is one of the fastest and low power consuming flip-flop designs [7], we designate it to be the baseline flip-flop. Fig. 1 shows the hardware implementation of the CCAFF. Compared to the traditional TGFF, the CCAFF introduces four additional transistors (the dotted area shown in Fig. 1) to perform the XOR function. Because using the D value would either increase the setup time or worsen the clock skew, we synthesize the $D \oplus Q$ function using only $\overline{D}$, Q and $\overline{Q}$ values. They are from nodes A, B and C, respectively.

Unlike the complementary CMOS logic, the CCAFF is a *pass-transistor logic* (PTL). It's well known that the major drawback of PTL is the *voltage offset* problem. (a) If $Q=0$, transistors *XN2* and *XP2* are nonconducting. Depending on the value of $\overline{D}$, x is equal to Q or $\overline{Q}$. There is no voltage offset in this case. (b) In contrast, if $Q=1$, the voltage offset would arise between Q and x via *XN1* (or between $\overline{Q}$ and x via *XP1*). To resolve this negative effect, we provide another level restored paths *p0* and *p1* from $\overline{D}$, as shown in Fig. 1. When $\overline{D}$ =1, x is equal to strong '*1*' which is restored from $\overline{D}$ via *XP2* (i.e., path *p1*). Similarly, x is equal to strong '*0*' which is restored from $\overline{D}$ via *XN2* (i.e., path *p0*), if $\overline{D}$ =0.

2.2 Timing, Power and Area Analyses

The basic timing parameters of a flip-flop design includes the clock-to-output propagation delay, setup time and hold time. From Fig. 1, since both the D input and *clk* pass through inverters before reaching T_1, any changes in the input after the clock goes high do not affect the output, i.e., the hold time is *0*. Thus, we only consider the clock-to-output delay and setup time in our design.

Clock-to-Output Delay: Refer to Fig. 1, the clock-to-output delays of both the CCAFF (t_{cq_CCAFF}) and TGFF (t_{cq_TGFF}) are determined by the same critical path from node Q_M to output Q through T_2, I_3, and I_4, which is equal to t_{pd_T2} + t_{pd_I3} + t_{pd_I4}. Because the CCAFF introduces four additional transistors to perform the value comparison, thus, to maintain the same clock-to-output delay as the TGFF, we have to resize the CCAFF to compensate the performance loss.

To resize the CCAFF in practice, a TGFF with minimum size and comparable performance had been implemented in the TSMC 0.18μm technology, in which the channel length is fixed at 0.18μm for all transistors, and the β ratios ($(W/L)_p/(W/L)_n$) of all the inverters are 1.5. Note that the output load capacitances C_Q and C_x are assumed to be 4 fF and 2 fF, respectively, in the following analyses. In the critical path t_{pd_T2} + t_{pd_I3} + t_{pd_I4}, we can enlarge the T_2 size to reduce the t_{cq_CCAFF}, but this would result in an increase in clock load, which worsens the clock skew problem. Thus, to reduce t_{cq_CCAFF}, our strategy is to minimize the delays of both inverters I_3 and I_4 as much as possible, except for the transmission gate T_2.

In our implementation, without *self-loading effect* the maximum sizing factors of inverters I_3 and I_4 are 1.2 and 1.8, respectively. Such enlargements are not enough to compensate the clock-to-output delay loss if the T_2 size of the CCAFF is the same as that used in the TGFF. Thus, we have to enlarge I_2 to help the reduction of the t_{cq_CCAFF}. From our measurement, when S_{I2}, S_{I3}, S_{I4} are 1.8, 1.2 and 1.8, the t_{cq_CCAFF} and t_{cq_TGFF} are almost the same value *0.183ns*.

Setup Time: Note that besides signal Q_M the result signal x must be also available before the rising edge of the clock in the CCAFF. Thus, the setup time of the CCAFF is given by: $t_{su\_CCAFF} = \max[t_{dm}, t_{dx}]$, which is different from the setup time of the TGFF (i.e., $t_{su\_TGFF} = t_{dm}$). Similar to the analysis of t_{cq}, the setup time would increase with the load capacitances of the inverters I_1 and I_2. Because the inverter I_2 has been resized for avoiding the lengthened t_{cq}, we only enlarge the I_1 to

Table. 1. The power consumption of a flip-flop operation in various write patterns.

Consuming Power (mW)	TGFF	CCAFF	*Increase*
0->0	7.54E-06	1.25E-05	*66.18%*
0->1	3.50E-03	6.31E-03	*80.29%*
1->0	3.78E-03	7.40E-03	*95.77%*
1->1	8.01E-06	1.35E-05	*68.54%*

compensate the setup time loss due to the increased load capacitances. Fig. 2 shows the t_{dm} and t_{dx} delays of the CCAFF in different sizing factor of inverter I_1. It is clear that t_{dx} is always less than t_{dm}, i.e., $t_{su\_CCAFF} = \max[t_{dm},t_{dx}] = t_{dm}$. The critical path of the CCAFF setup time is still t_{dm}. From Fig. 2, when S_{I1} is 2.6, the setup times are of the same *0.162ns* for both the TGFF and CCAFF.

Area Cost: Based on the analyses described above, to maintain the same clock-to-output delay and setup time as the TGFF, in the CCAFF we have to resize the I_1, I_2, I_3 and I_4 by 2.6, 1.8, 1.2 and 1.8 times, respectively. Therefore, the penalties of both power consumption and area cost are unavoidable. Compared to the TGFF, the CCAFF area is increased from *157.42μm²* to *231.76μm²*. Most area overhead is introduced by the large inverters and the additional transistors used in comparison logic that imposes around a *47.2%* area overhead.

Power Consumption: From our measurement, Table 1 shows the flip-flop power consumption for various write patterns. Due to no state transition, the power consumed in the 0->0 and 1->1 write patterns are much less than that in the 0->1 and 1->0 write patterns. Compared to the conventional TGFF, the CCAFF would increase the power consumption by *66%~96%*.

2.3 Apply CCA Register to the TLB

Fig. 3 illustrates our framework for the real-time detection of a block buffer hit. From Fig. 3, the tag field of the address is used to determine whether the desired mapping is held in the TLB. To detect the block buffer hit as soon as the address is generated, instead of the traditional TGFF we use the proposed CCAFF to implement the tag field. The tag length depends on the TLB configuration. For example, the tag field is the high-order 20 bits (b_{31}-b_{12}) for a fully-associative TLB with 4KB page size.

For noise prevention, the additional outputs generated by the CCAFF are connected to a pseudo-NMOS NOR logic to produce the result signal *TG* (tag change) to control the TLB lookup. Because the capacitance of the pseudo-NMOS NOR logic has been considered in the timing analyses provided in Section 2.2 (C_x is assumed to be 2 fF), it would not impair the performance of the CCAFF. During the clock low, if the coming memory reference and the current one have different tag, at least one of the signals x_0-x_q would be 1 such that the input of the indicator TGFF is 0. At the rising edge of the clock, the address register would be updated while the signal *TG*=1 indicates there is a tag change between the coming

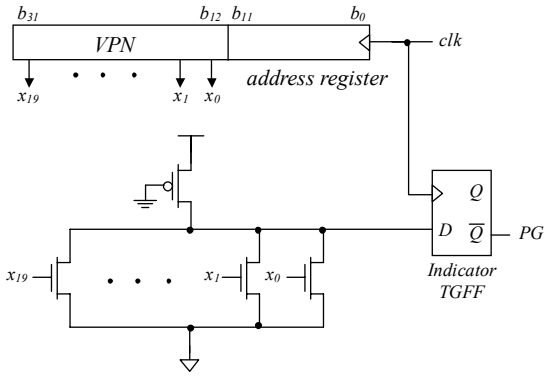

Fig. 3. A 32-bit CCA address register and the peripheral circuitry for the real-time detection of a block buffer hit.

reference and the current one. Thus, the TLB lookup is necessary. In the other case, if the coming reference and the current one have the same tag (i.e., no tag change), all of signals x_0-x_q remain 0 such that *TG*=0. It is used to disable the TLB lookup, and the corresponding data could be delivered directly from the block buffer.

3. Adaptive Variants of Banked TLB

The key idea of the second technique is to reduce the power consumption per TLB access in the case of block buffer miss, so the power dissipated in TLB accesses can be further reduced.

3.1 Banked TLB

The *banking technique* [5] is a power efficiency design that divides the entire TLB space into several smaller banked TLBs. Because only a part of the entries are looked up, the banked TLB consumes less power than a fully-associative TLB on each access. Due to the restricted entry allocation, however, the major drawback of the banked TLB is that the capacity misses would tend to increase so that it has a negative effect on the performance. To alleviate the performance penalty introduced by the banking technique, we have developed two adaptive variants of the banked TLB to reduce the power consumption per access without increasing the miss ratio.

3.2 The First Variant of Banked TLB Design

Similar to the conventional banked TLB, in the first variant of banked TLB design (called *VB1* TLB), we partition the entire TLB into several smaller banks. All banks are fully-associative and have the same entry number. The *n*-way banked TLB means that a TLB is partitioned into *n* banks. For example, Fig. 4(a) shows a 4-way banked TLB. By banking the TLB, ideally, the power consumption can be approximately reduced to 1/*n* in accessing a *n*-way banked TLB. The simplest and well-known banking function is the *bit selection*, in which some tag bits of the virtual address are used as the selection bits to activate a specific bank, called *target* bank. Although the bit selection banking function

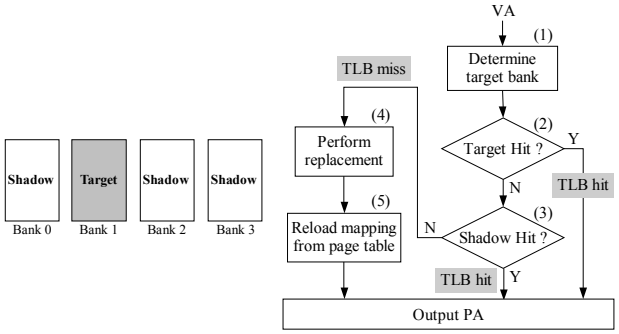

Fig. 4. A 4-way VB1 TLB design. (a) is the block diagram. The gray block symbolizes active bank. (b) is the access flow.

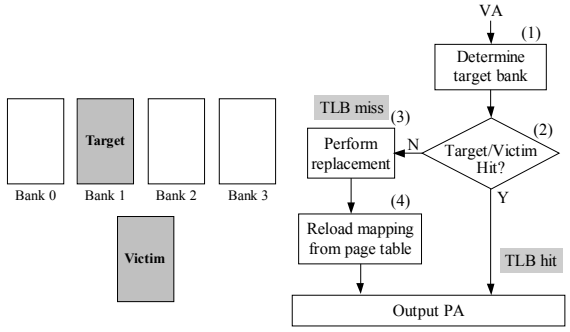

Fig. 5. A 4-way VB2 TLB design. (a) is the block diagram. The gray blocks symbolize active banks. (b) is the access flow.

cannot achieve an optimal entry distribution among these banks, due to the simplicity it does not increase the access time. Consequently, we simply use the least significant tag bits to determine the target bank throughout this paper.

Allocation Policy for the VB1 TLB: In the VB1 TLB, we first define those that are not target are *shadow* banks. The difference between our VB1 and the conventional banked TLBs is the allocation policy. In the VB1 TLB, when a miss occurs, we first check whether the target bank contains available entries. If there is a free entry, then we allocate it to the incoming page mapping. Otherwise, a replacement algorithm is initiated on the target bank to replace an entry in order to make a room for the incoming page mapping. Unlike the conventional banked TLB, in which the entry replaced from the target bank (called *victim entry*) is discarded directly, the VB1 TLB removes the victim entry to the shadow bank. This is because the replaced data has already been fetched, it can be used again at small cost. Note that if the shadow banks have no free entry, we have to discard an entry from the shadow bank for accommodating the victim. Because the random method has comparable performance and easy implementation, the replacement used in the VB1 TLB design is the random method.

From the description of the allocation used in VB1 TLB, we conclude the most distinct features of the VB1 TLB as follows. (1) The desired page mapping may be held in either the target or shadow banks, so that there are two possible hit cases: *target hit* and *shadow hit*. (2) With the same entry number, the miss ratio of the VB1 TLB is actually identical to that of the fully-associative TLB. It can be verified by the experimental results Fig. 6 shown in Section 4. In other words, the VB1 TLB combines the high power efficiency of the conventional banked TLB and the low miss ratio of the fully-associative TLB. The entire access flow of the VB1 TLB is illustrated in Fig. 4(b) and described as follows:

(1) Use the least significant tag bits of virtual address (VA) to determine the target bank.
(2) If the desired page mapping is found in the target bank, it is a *target hit*. The physical address (PA) is output to accomplish the TLB access.
(3) In case of target miss, if the desired page mapping is found in the shadow bank, it is a *shadow hit*. Besides generating the PA, we have to choose a victim entry from the target bank to swap the hit entry in the shadow bank, because it is likely a target hit in the following accesses.
(4) In case of both target and shadow misses, it is a TLB miss. We then have to perform the entry allocation described previously.
(5) Reload the desired page mapping from the page table to output the PA.

From Fig. 4(b), because the shadow hit must occur after the target miss, besides consuming more power, the shadow hit also increase the access time to impair the performance. Thus, in our VB1 TLB design the target hit rate is the major lever on both the power and performance efficiency.

3.3 The Second Variant of Banked TLB Design

Despite the VB1 TLB with the same miss rate as the conventional fully-associative TLB, from the access flow Fig. 4(b) one performance weakness is that the target and shadow banks are looked up in serial. Consequently, we propose the second variant of the banked TLB, called *VB2* TLB, which aims to avoid accessing the TLB twice for one hit. As shown in Fig. 5(a), in the VB2 design we partition the entire TLB into a set of normal banks and a victim bank. The size of victim bank can be unequal to that of normal bank, but all normal banks must have the same size. Borrowed from the concept of the victim cache [8], the victim bank is shared by all normal banks and contains only the replaced entries that are discarded from the normal banks. Similar to the VB1 TLB, we use the random replacement and the simple bit selection to determine the target bank in the VB2 design.

Allocation Policy for the VB2 TLB: Compared to the VB1 design, the difference is that the VB2 has distinct allocation policy in case of target miss. Unlike the VB1 TLB, in which the victim entry is removed to the shadow bank, we remove the victim entry to the victim bank in the VB2 TLB. Note that if the victim bank has no free entry, we have to discard an entry for the victim from target bank. The VB2 design has also two hit cases: *target hit* and *victim hit*. The entire access flow of the VB2 TLB is illustrated in Fig. 5(b).

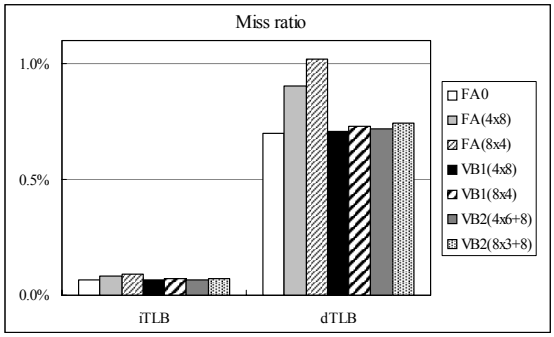

Fig. 6. Miss ratios for all evaluated TLBs.

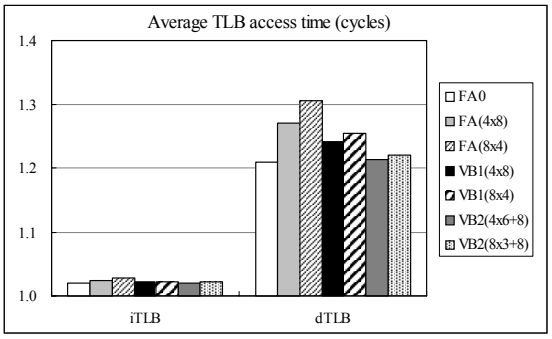

Fig. 7. Average TLB access times for all evaluated TLBs.

From this figure, all steps are identical to the VB1 design except for the Step 2, in which the target and victim banks are looked up in parallel. If the desired page mapping is found in the target bank, it is a *target hit*. The physical address (PA) is output to accomplish the TLB access. If the desired page mapping is found in the victim bank, besides the generation of PA, we have to choose a victim entry from the target bank to swap the hit entry in the victim bank, because it is likely a target hit in the following accesses. The VB2 can alleviate the performance penalty incurred by the VB1 design, but two banks, i.e., the target and victim banks, are always activated on each access. This implies that the parallel accessing of the VB2 comes at the cost of more power consumption per access than the VB1.

4. Experimental Results

In this paper we use *SimpleScalar* toolset to model a baseline processor with split instruction TLB (iTLB) and data TLB (dTLB). Both they are fully-associative with 32-entry. To avoid an explosion in the number of simulations, the *page size* is fixed to be 4K. The input benchmark is a subset of *SPEC2000*. There are four essential evaluation metrics used in this paper. They are miss ratio, average TLB access time, average energy consumption per TLB access, and Energy*Delay product. In our comparison, FA0 is the conventional fully-associative TLB with a block buffer. FA(m×n) and VB1(m×n) denote the fully-associative TLB implemented as the conventional m-way banked and our m-way VB1 design, respectively, in which each bank contains n-entry. VB2(m×n+p) is the VB2 TLB that consists of m banks (with size of n-entry) and a p-entry victim bank. For a fair comparison, note that the entry number for all the evaluated TLBs is always 32-entry even in the VB2 design. Several experiments were performed to conclude that the optimum entry number of victim bank used in VB2 TLB should not be less than 8.

4.1 Miss Ratio

Fig. 6 shows the miss ratios for all TLBs. Because the difference between integer and floating-point programs is hardly noticeable, we do not present these two benchmarks separately in the following discussion. From this figure, the conventional banked TLB would largely increase the miss ratio, especially for the dTLB. In contrast, the VB1 design has almost the same miss ratio as the unbanked FA TLB. Because the available allocation space of the VB2 is smaller than that of the VB1, the miss ratio of the VB2 is slightly higher than the VB1 miss ratio. One common trend for these banking techniques, including VB1 and VB2, the miss ratio would increase with the banking degree.

4.2 Average TLB Access Time

For the conventional banked, VB1 and VB2 TLBs, the time to access the target bank is assumed to be one cycle. Particularly, in the VB1 TLB, if a target miss occurs, then we have to pay an additional cycle to look up all the shadow banks. Thus, the time required for a shadow hit is two cycles. The miss penalty for a miss handling interrupt routine is assumed to be 30 cycles, which is based on values for 32-bit StrongARM processor [9]. Fig. 7 shows the average access times for all the evaluated TLBs. Due to the lower miss ratio, our designs have superior average access time than the conventional banked TLB, especially for the dTLB with poor spatial locality.

4.3 Average Energy Consumption per TLB Access

To evaluate the average energy consumption per TLB access, we simply accumulate the energy dissipated on each access, and then finally divided by the number of total accesses. For all the evaluated TLB designs, the average energy consumption per access are given by the following equations (1)~(4), respectively:

$$AE_{FA0} = \left(\sum E(BB) + \sum E(TLB_{FA}) + \sum E(miss)\right) / Accesses \quad (1)$$

$$AE_{FA} = \left(\sum E(BB) + \sum E(target) + \sum E(miss)\right) / Accesses \quad (2)$$

$$AE_{VB1} = \left(\sum E(BB) + \sum E(target) + \sum E(shadow) + \sum E(miss)\right) / Accesses \quad (3)$$

$$AE_{VB2} = \left(\sum E(BB) + \sum E(target) + \sum E(victim) + \sum E(miss)\right) / Accesses \quad (4)$$

In Eq. (1), $E(TLB_{FA})$ is the energy dissipated in accessing the unbanked fully-associative TLB. Because the target bank size of the VB2 is smaller than that of the VB1, in Eq. (4) the term $E(target)$ is not equal to $E(target)$ shown in both Eq. (2) and (3). In these equations, except for the $E(BB)$ and $E(miss)$, all the energy values can be estimated by using *CACTI* tool [10] configured with 0.18μm technology. By

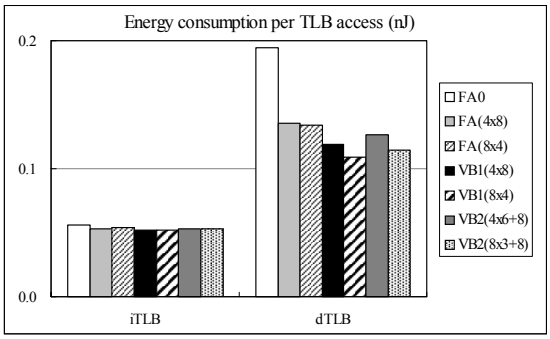

Fig. 8. Energy consumption per TLB access for all TLBs.

performing the *HSPICE* simulation of the extracted layout, we obtain the energy dissipated in the block buffer hit, i.e., *E*(*BB*), is about 0.046 nJ, and refer to [6], the energy dissipated in a TLB miss, i.e., *E*(*miss*), is about 8.85 nJ.

Fig. 8 shows the average energy consumption per access for all TLB structures. Because the iTLB has very strong spatial locality, the difference between the conventional banking and our designs is hardly noticeable. In contrast, for the dTLB, our design is better than the conventional banked TLB in reducing the average energy consumption per access. For example, compared to the FA(4×8), with the same banking degree the VB1(4×8) can achieve about 12.5% improvement.

4.4 Energy*Delay Product

Combine the results of average access time and energy consumption per access, we can obtain the Energy*Delay (ED) product, as shown in Table 2. The small ED product is the goal of our design. Note that, for the same banking degree, the VB1 and VB2 TLB almost have the same ED product. For the iTLB, our design improves the ED product of the FA0 roughly by 1.97% and 4.11% for 4-way and 8-way banking structures. The improvement is not significant. In contrast, for the dTLB, our design can reduce 13.75% and 20.95% ED product of the FA0 for 4-way and 8-way banking structures, respectively. This result confirms that our design is particularly beneficial for the TLBs with poor locality.

5. Conclusions

In this paper, we propose the first technique, *real-time filter*, to reduce the number of TLB accesses, and then propose the second technique to reduce the power consumption per TLB access. Strictly, the first technique can be regarded as an alternative implementation to the block buffering, which removes the comparison delay penalty from the conventional block buffering. The second technique aims to reduce the high miss ratio incurred by the conventional banking. The experimental results conclude that our design shows superior power efficiency than the conventional banked TLB.

Table. 2. Energy*Delay products for all evaluated TLBs.

Energy*Delay	iTLB	dTLB
FA0	0.0568	0.2351
FA(4x8)	0.0549	0.1730
FA(8x4)	0.0557	0.1743
VB1(4x8)	0.0538	0.1492
VB1(8x4)	0.0535	0.1378
VB2(4x6+8)	0.0546	0.1537
VB2(8x3+8)	0.0544	0.1405

References

[1] I. Kadayif, A. Sivasubramaniam, M. Kandemir, G. Kandiraju and G. Chen, "Generating Physical Address Directly for Saving Instruction TLB Energy," in Proc. of International Symposium on Microarchitecture, 2002, pp. 185-196.

[2] T. Juan, T. Lang and J. Navarro, "Reducing TLB Power Requirements," in Proc. of International Symposium on Low Power Electronics and Design, 1997, pp. 196-201.

[3] H. Levy and R. Eckhouse, "Computer Programming and Architecture, The VAX, " Digital Press, 1989.

[4] J. Kin, M. Gupta and W. H. Smith, "The Filter Cache: An Energy Efficient Memory Structure," in Proc. of International Symposium on Microarchitecture, 1997, pp. 184-193.

[5] S. Manne, A. Klauser, D. Grunwald and Somenzi, "Low Power TLB Design for High Performance Microprocessors," Univ. of Colorado Technical Report, 1997.

[6] J. H. Choi, J. H. Lee, S. W. Jeong, S. D. Kim and C. Weems, "A Low Power TLB Structure for Embedded Systems," IEEE Computer Architecture Letters, Vol. 1, 2002

[7] R. Ramanarayanan, N. Vijaykrishnan and M. J. Irwin, "Characterizing Dynamic and Leakage Power Behavior in Flip-Flops," 2002, pp. 433-437.

[8] N. P. Jouppi, "Improving direct-mapped cache performance by the addition of a small fully-associative cache and prefetch buffers," in Proc. of International Symposium on Computer Architecture, 1990, pp. 364-373.

[9] R. Witek and J. Montanaro, "StrongARM: A High-Performance ARM Processor," in Proc. of COMPCON, 1996, pp. 188-191.

[10] P. Shivakumar and N. P. Jouppi, "CACTI 3.0: An Integrated Cache Timing, Power, and Area Model," COMPAQ WRL Research Report, 2001.