가상주소 변환 과정에 대한 부담의 줄임

우 종 정†

요 약

메모리의 계층적 구조는 메모리의 접근 속도를 개선하고 프로그램 공간을 확장하는데 유용한 메카니즘이다. 그러나 이 구조는 데이타의 참조를 위해서 적어도 두번-주소 변환을 위한 TLB 와 원하는 데이타를 위한 데이타 캐시-의 메모리 접근이 필요하다. 만약 캐시의 크기가 가상 메모리의 페이지 크기와 캐시 메모리의 연관 정도의 곱보다 커지면 TLB접근과 데이타 캐시의 접근을 병렬로 수행하기 어려우며, 따라서 프로세서 타이밍의 임계 경로가 길어져 성능에 영향을 미친다. 이들의 병렬 접근을 성취하기 위하여 직접 사상 TLB와 조그마한 완전 연관 사상 TLB를 결합한 혼합 사상 TLB를 제안한다. 전자는 TLB 접근에 따른 지연시간을 줄일 수 있으며 후자는 전자로부터 발생한 충돌 부재를 제거할 수 있게 된다. 트레이스 구동 모의 실험 결과에 의하면 제안된 TLB는 4개의 엔트리로만 구성된 완전사상 TLB를 추가하더라도 부재율의 상승에 의한 영향이 주소변환에 따른 지연시간 축소에 의하여 상쇄되므로 효과적이다.

Reducing the Overhead of Virtual Address Translation Process

Jongjung Woo†

ABSTRACT

Memory hierarchy is a useful mechanism for improving the memory access speed and making the program space larger by layering the memories and separating program spaces from memory spaces. However, it needs at least two memory accesses for each data reference : a TLB(Translation Lookaside Buffer) access for the address translation and a data cache access for the desired data. If the cache size increases to the multiplication of page size and the cache associativity, it is difficult to access the TLB with the cache in parallel, thereby making longer the critical timing path in the processor. To achieve such parallel accesses, we present the hybrid mapped TLB which combines a direct mapped TLB with a very small fully-associative mapped TLB. The former can reduce the TLB access time, while the latter removes the conflict misses from the former. The trace-driven simulation shows that under given workloads the proposed TLB is effective even when a fully-associative mapped TLB with only four entries is added because the effects of its increased misses are offset by its speed benefits.

1. Introduction

The processing power of a computer is closely related to the speed and the capacity of its memory system. An ideal memory system should provide a very short access time as well as a very large address space. To achieve these ends at a reasonable cost, most computers today rely on cache memory and on virtual memory. Cache memory provides a fast and effective memory access time by exploiting the property of locality. Its effectiveness is primarily dependent on its access time and its miss ratio. On the other hand, virtual memory provides users with the appearance of a memory with as large as the swapping space and as fast as the physical

· This Paper was supported by the Grants for Professors of Sungshin Women's University in 1995.

† 정 회 원 : 성신여자대학교 조교수

논문접수 : 1995년 9월 20일, 심사완료 : 1995년 11월 30일.

memory. Each data reference in the memory hierarchy system requires at least two transactions: a TLB access and a cache access, thus affecting the processor cycle time which is one of the most important factors in recent microprocessors. Reduction in the processor cycle time provides the improvement of the system performance, given the same architecture. Therefore, despite having worse miss ratios, direct mapping becomes popular in the current cache because as the cache size increases, direct mapping provides data references faster [1]. But, if real address caches which are direct mapped and larger than a page size are employed with traditional address translation, address translation can not be performed in parallel with cache accesses. In order to achieve this, we use a hybrid mapping in the TLB. Hybrid mapping combines direct mapping and fully-associative mapping in a single memory. The hybrid mapped TLB consists of a relatively large direct mapped TLB and a relatively small fully-associative mapped TLB. This idea was proposed for the caches by both J. Woo [2] and N. Jouppi [3]. They showed that their proposed caches reduced the miss rate a lot on their benchmarks. This idea goes further by applying to the TLB, thereby reducing the overhead of the address translation process. This paper shows that the hybrid mapped TLB can eliminate the address translation latency from the processor's critical path without deteriorating the overall performance even when a small fully-associated mapped TLB is added.

2. The Overheads of Address Translation Process

In hierarchical memory systems, the location of the memory management unit(MMU) is an important design consideration. Since cache memories are also located between the processor and main memory, there are two possible locations for the TLB. If the MMU is located in between main memory and the cache, the cache should be indexed with the virtual addresses. It is called a virtual address cache, while the other is a real address cache. The processor generates virtual addresses but the data are stored in main memory based on real addresses. The MMU interfaces between processors and main memory by translating virtual addresses into real addresses. Hence, we assume that the MMU performs address translation on a page basis without loss of generality. In addition, we discuss only the data TLB because the instruction TLB generally exhibits good performance [4].

For the real address cache, a TLB access must occur before a cache access. The address translation consists of the effective address calculation and the TLB access. The contemporary RISC processor is typically implemented with five basic execution steps: instruction fetch(IF), instruction decode and register fetch(ID), execution and effective address translation(EX), memory access (MEM), and write back(WB) [5]. The TLB access is typically performed in the MEM pipeline stage, which can become the slowest of all pipeline stages in the processor. When the cache size is relatively small, this problem is not critical. The reason is that the cache access(MEM stage) can be performed in parallel with the TLB access because the cache can be indexed only by the page displacement which remains invariant across the address translation process. These days, with advances in VLSI technology, the cache size tends to grow larger than the page size. If the cache is larger than the multiplication of

the page size and the cache associativity, caches cannot be indexed only by the invariant field of the virtual address in the conventional TLB, which makes the MEM stage the slowest. There are several approaches to eliminate the TLB access latency : using virtual address caches, making the page size larger, increasing the cache associativity, and dividing the stage of memory access pipeline.

Virtual address caches are attractive because they can be accessed without address translation. Unfortunately, they have lots of disadvantages [5, 6, 7, 8]. Two distinct virtual addresses are mapped onto the same physical address and the same virtual address can be mapped onto distinct physical addresses, which is called the synonym problem. It also complicates keeping protection information with the data cache because the page protection bits and other fields need to be included in each cache block of a virtual address cache. In addition, it has to flush the cache on process switches. Larger pages make the page displacement wider. Thus we can use large caches for parallel address translations. In the DEC MultiTitan, the large page size is used for this purpose [3]. However, large pages make the protection granularity coarser and memory fragmentation problem worse because the pages can be larger than what the program needs. Higher cache associativity makes the cache slower but in IBM 3081, a high degree of associativity is used for increasing the cache [9]. Finally, increasing the number of memory access pipeline stages provide several problems such as clock skew and latch overhead, longer load/store, and slower branches [8].

There are other alternatives for improving TLB performance : supporting multiple page sizes [10], using base address caches [5], and the TLB slices [11]. The first may not eliminate TLB latencies and even make memory management and page replacement policies complex. The second needs an extra base address caches with more than 32 entries even if it can hide TLB access latencies. The last can work well only for the second level cache if the second level cache is shielded by a virtual indexed primary cache.

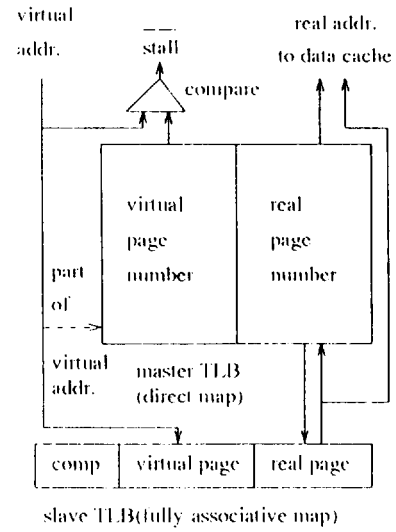
3. Hybrid Mapped TLB

Mapping policies are usually classified according to their associativity. For a given size, a higher degree of set-associativity generally yields a better miss ratio but leads to a degradation of effective memory access time, while a lower degree of set-associativity yields a reduction of effective memory access times but leads to an increase of cache miss ratios [1,2,3,6,8]. Owing to the above conflicting relation, the traditional caches will not be able to keep up with the speeds of modern microprocessors. To improve both miss ratio and access time at the same time, the concept of hybrid mapped caches combining two differently mapped caches in a single cache was proposed [2, 3]. One, called a master, is large and direct mapped, while the other, called a slave, is relatively small and fully-associative mapped. Since the master is direct mapped, it is typically cheaper and faster;but its hit ratio is generally not better than the hit ratios of the other mapping schemes because a later retrieved entry is more likely to be discarded in direct mapping than in other mapping. To compensate for these conflict misses, a small size slave is employed and it contains the entries displaced from the master. Note that hybrid mapping does not need to adhere to the "inclusion property" because there are no data which belong to both the master and the slave.

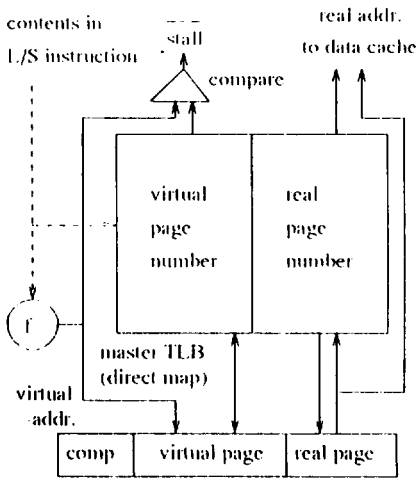
The TLB is a special cache for the virtual address translation. Modern microprocessors typically make use of fully-associative mapped TLBs in order to obtain low miss ratios [12, 13]. But such TLBs provide relatively long access times. Henceforth, by applying the hybrid mapping policy for virtual address translation, we can eliminate or hide the time for virtual address translation, thereby reducing the critical path of accessing the real address cache. Especially, hybrid mapping is more useful for data references because more conflict misses of data are eliminated by the slave cache than those of instructions [3].

(Figure 1) illustrate two organizations of hybrid mapped TLBs; one is indexed by part of the virtual page number and the other is by partial contents of the load/store instruction such as its base register identifier [5, 14, 15] and its reference offset. The partial contents of the load/store instruction are attractive because the TLB can be accessed before the calculation of the virtual page number. However, the effectiveness of partial contents of the load/store instruction as a TLB index relies on how rarely the base register is modified. In this case, the virtual page number retrieved from the master TLB is different from the the virtual page number calculated by the processor when the base register is modified. Note that the traditional TLB can only be indexed by the full contents of the virtual page number unlike the hybrid mapped TLB. From the speed point of view, the master TLB has some improvement factors as below:

- Since the master TLB is direct mapped, in most cases it is faster than traditional TLBs.
- When the master TLB is indexed by the partial contents in the load/store instruction, the TLB access can be performed before the MEM stage.
- When part of the virtual page number is employed as a TLB index, the time for comparing the virtual page number in the master TLB with the incoming virtual page number from the processor can be hidden because of the following reason. Generally



(a) with virtual page number



(b) with L/S instructions

(Fig. 1) Hybrid Mapped TLBs

the time for TLB references consists of the time for two operations: directory comparison and appropriate entry access. In the traditional TLB, those two operations should be performed in the forward sequence, namely an entry access after a directory comparison. The reason for this access order is because the appropriate entry access is dependent on the result of the directory comparison. On the other hand, in the direct mapped TLB, its operations can be reversed for decreasing the effective cache access delay. As a result, the operation of TLB directory comparison can be overlapped with the cache reference.

Therefore, if the hit ratio of the master TLB is reasonably high, the effective access time of the hybrid mapped TLB is shorter than the traditional TLB. From the space and cost point of view, fully-associative searches from translation tables are costly in terms of the logic gate count. Given the same number of hybrid mapped TLB entries and traditional TLB entries, the smaller silicon area is used in the hybrid mapped TLB rather than in the traditionally mapped TLB because of a smaller number of directory comparators required in the master TLB.

The desired data are obtained through two steps : a virtual address translation and a cache access. We assume that the TLB is always hit on the master TLB or the slave TLB because the miss ratios of hybrid mapped TLBs are almost negligible as shown in Section 4.2. If a miss is encountered on the hybrid mapped TLB, its procedure is almost the same as the procedure of traditional address translation. The followings are the procedures of the data reference operation for the hybrid mapped TLBs.

TLB indexed by the virtual page number

(1) Using a subset of virtual page bits, only one page table entry is read out from the master TLB. Each entry in the master TLB consists of a virtual page number (V_{Tm}) and a real page number (R_{Tm}). At the same time, for the slave TLB, all virtual page numbers (V_{Ts}) are read out.

(2) The cache can be accessed by using R_{Tm} and the page offset regardless of the validity of R_{Tm}. In parallel, the R_{Tm} is compared with the incoming virtual address. For the slave TLB, all V_{Ts} are also compared with the incoming virtual address.

(3) If the R_{Tm} matches the incoming virtual address, the cache access by R_{Tm} is valid; otherwise, the processor stalls the pipeline for one cycle, while another cache access can be initiated by using the R_{Ts}, if the slave TLB is hit.

TLB indexed by the partial contents of the load/store instruction

(1) Using the partial contents of the load/store instruction, only one page table entry is read out from the master TLB. In the meantime, the virtual address is calculated from the base register and the reference offset.

(2) The cache can be accessed by using R_{Tm} and the page offset regardless of the validity of R_{Tm}. In parallel, the R_{Tm} is compared with the calculated virtual address. For the slave TLB, all tags (V_{Ts}) are read out.

(3) If the R_{Tm} matches with the calculated virtual address, the cache access by R_{Tm} is valid; otherwise, the processor stalls the pipeline for one cycle. In parallel, for the slave TLB, all (V_{Ts}) tags are

compared with the calculated virtual address.

(4) If any tag in the slave TLB matches with the calculated virtual address, the cache access is performed by R^{TS}.

If both TLBs are missed, the processor is stalled and initiates them to reload the missing page table entry. Note that if there is a miss on the master TLB and a hit on the slave TLB, the access penalty is just one cycle and if both TLBs are missed, the access penalty is the same as the conventional penalty.

4. Performance Analysis

The trace-driven simulation is a powerful tool which allows the study of several aspects of computer design and performance [6]. We used trace-driven simulation in order to examine the effectiveness of the proposed TLB. The traces were collected on a Sun Sparc workstation using the Spa package, which interprets the program execution to produce address traces. We sampled the first one billion instruction traces if the benchmark takes more traces than one billion traces for each cases. These sizes of traces can be sufficient to reduce the effect of compulsory misses. The sampled traces are passed into the TLB simulators through the UNIX pipeline without storing them. The LRU replacement policy is assumed for the slave TLB, which does not make the implementation expensive because their sizes are too small. The workloads we used are SPEC benchmarks (release 1).

4.1 Performance Metrics

The most commonly used metric of TLB performance is miss ratio, the probability which the desired information is not found in the TLB. But it might be misleading because the effects of cycle stalls caused by TLB misses for data are different from those of cycle stalls caused by cache misses for instructions. Given an instruction set, a performance metric that is directly related to the average cycles per instruction (CPI) is the misses per total executed instructions, MPI [16]. The equation for MPI can be written as,

$$\text{MPI}=\frac{\text{total number of misses}}{\text{total number of instructions executed}} \quad \cdots\cdots (1)$$

Another commonly used metric for TLB performance is effective access time, the average latency time between when the processor requests a page table entry and when it receives that page table entry. But since it has a similar problem to the miss ratio, we introduce yet another performance metric, the average time of the underlying pipeline stage (TPS). The TPS can be expressed as follows:

$$\overline{t_{pipe}(p)}=n\times t+\text{MPI}\times t \quad \cdots\cdots (2)$$

where n, $t_{pipe}(p)$, t_c and t_p are the number of cycles per instruction with no TLB misses, the average time of p stage, the processor cycle time, and the average cache penalty, respectively. Our relevant pipeline stage is the memory access(MEM) stage of the pipeline for the data.

4.2 Simulation Results and Analysis

To demonstrate the effectiveness of the hybrid mapped TLB, we analyze the effects of cycle stalls caused by misses of the master TLB. If such effects are not offset by the speed benefits of the master TLB, our scheme must be advantageous.

〈Table〉 MPI for Master TLBs

page \ TLB	16	32	64	128	256
4K	.08425	.05319	.03229	.01203	.00670
8K	.08653	.04178	.02012	.00746	.00080
16K	.09024	.04243	.02093	.00121	.00000
32K	.05033	.02395	.01127	.00065	.00003

〈Table 2〉 MPI for Hybrid Mapped TLBs and Traditional TLBs

page \ TLB		20	36	68	132	260
4KB	hybrid	.00106	.00037	.00012	.00006	.00004
	ssoc	.00062	.00019	.00006	.00003	.00000
8KB	hybrid	.00048	.00017	.00006	.00003	.00000
	assoc	.00025	.00006	.00002	.00000	.00000
16KB	hybrid	.00032	.00007	.00003	.00001	.00000
	assoc	.00010	.00002	.00000	.00000	.00000
32KB	hybrid	.00010	.00005	.00001	.00000	.00000
	assoc	.00003	.00000	.00000	.00000	.00000

• The slave TLB of hybrid mapping consists of four entries.

〈Table 1〉 shows the weighted arithmetic average MPIs[1] of the master TLBs for the different page sizes (4~32K bytes long) and number of TLB entries (16~256). This table shows that the use of larger pages or more TLB entries can significantly improve the performance of TLB. Like 〈Table 1〉, 〈Table 2〉 shows the average MPI of the hybrid mapped TLBs and the traditional TLBs. The hybrid mapped TLBs are composed of the master TLBs in 〈Table 1〉 and the slave TLBs with four entries to permit a fair comparison because the hybrid mapped TLBs and the traditional TLBs have the same number of entries. As shown in 〈Table 1〉 and 2, only four TLB entries of the slave TLB are sufficient to eliminate most conflict misses of the master TLB because their differences are mostly less than 0.01 %. According to Eq.(1) and (2), we can express the average time of the memory access pipeline stage for the traditional TLB, $\overline{t_o\ (MEM)}$, as below :

[1] The total number of misses divided by the total number of simulated instructions for given workloads.

$$\overline{t_o\ (MEM)} = n \times t_C\ (\text{TLB}\ _o) + \text{MPI}(\text{TLB}\ _o) \times t_P \quad \cdots\cdots (3)$$

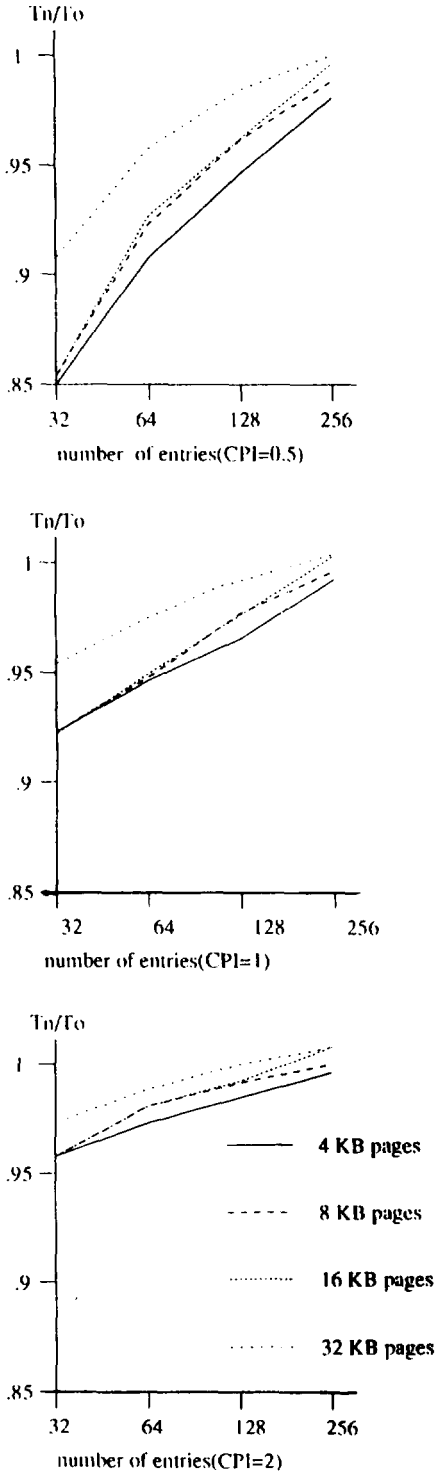
where MPI(TLB $_o$) denotes the MPI for the given traditional TLB organization. In a similar way, the average time of the memory access pipeline stage for the hybrid mapped TLB, $\overline{t_n\ (MEM)}$, can be given as :

$$\overline{t_n\ (MEM)} = (n + \text{MPI}(\text{TLB}\ _m)) \times t_C(\text{TLB}\ _n) + \text{MPI}(\text{TLB}\ _n) \times t_p \quad \cdots\cdots (4)$$

Here MPI(TLB $_m$) and MPI(TLB $_n$) refer to the MPI for the given master TLB and the hybrid mapped TLB. From both equations, we can ignore the second item because MPI(TLB $_o$) and MPI(TLB $_n$) are too small as shown in 〈Table 2〉. Thus, compared to Eq.(3) and (4), the processor cycle time of the hybrid mapped TLB can be justified if the following equation is satisfied:

$$t_C(\text{TLB}\ _n) \le \frac{n}{n + \text{MPI} + (\text{TLB}\ _m)} \times t_C(\text{TLB}\ _n) \quad \cdots\cdots (5)$$

Here we call $n/(n+\text{MPI}(\text{TLB}\ _m))$ the maximum effective cycle time ratio of the hybrid mapped TLB. Therefore, the usefulness of the proposed TLB scheme relies largely upon the MPI of the master TLB if given n. (Figure 2) shows three graphs which illustrate the relationship between the maximum effective cycle time ratio and the number of master TLB entries for three n's(0.5, 1.0, and 2.0). These graphs reveal that for typical RISC microprocessors (CPIs≈1), the maximum effective cycle ratios are generally about 0.9835 for 128 TLB entries, 8 K byte pages, and CPI=1.2. In other words, the proposed scheme can be justified if t_C (TLB $_n$) is smaller than $0.9835 \times t_C$ (TLB $_o$). This time reduction can be achieved by reducing TLB access time and hiding the comparison latency of the virtual page number in the master



(Fig. 2) Maximum Effective Cycle Ratio

TLB with the incoming virtual page number from the processor. Furthermore, if the number of master TLB entries is larger than 64, the proposed scheme can be effective for the superscalar processors (CPIs(1) because even a small amount of cycle time improvement can be justified. Therefore, we can tell that for real-addressed large caches, the address translation latency can be reduced by employing the hybrid mapped TLB scheme with little extra overhead or die size.

5. Conclusion

In most processor designs, the cache access (real address caches) forms the critical path of the execution pipeline, thereby determining the processor cycle time [8]. To reduce the effect of the TLB on its critical path, we present a hybrid approach to TLB mapping which combines a direct mapping with a fully-associative mapping. In a hybrid mapped TLB, most TLB hits are achieved in the master TLB and most conflict misses of the master TLB are removed by the small slave TLB. Thus, the master TLB provides fast access times, and the slave TLB assists to eliminate the most conflict misses. Furthermore, since the master TLB can perform even its tag comparison with its underlying cache access in parallel, its performance can be significantly improved for data references. Especially, the hybrid mapped TLB can be very desirable if it is indexed by partial contents of the load/store instruction because the TLB is accessed not at the memory access pipeline stage but at an earlier stage.

Our simulation results show that most conflict misses of the master TLBs are removed by their slave TLBs, even only four slave TLB entries. The hybrid mapped TLB is appropriate for the typical microprocessor if the

effective cycle ratio is under 0.9835 for the 128 TLB entries and 8 K byte pages. We can completely hide the TLB access latency when the TLB is a hit in the master, thereby reducing the processor cycle time.

References

[1] M. Hill, "A Case for Direct-Mapped Caches," Computer, pp.25-40, Dec. 1988

[2] J. Woo, "Hybrid Cache Mapping," Univ. of Texas at Austin, Thesis, May 1990

[3] N. Jouppi, "Architectural and Organizational Tradeoffs in the Design of the MultiTitan CPU," The 15th Int'l Symp. on Computer Architecture, pp. 281-289, 1988

[4] J. Chen, A. Borg, and N. Jouppi, "A Simulation Based Study of TLB Performance," The 19th Int'l Symp. on Computer Architecture, pp. 114-123, May 1992

[5] T-C. Chiueh and R. Katz, "Eliminating the Address Translation Bottleneck for Physical Address Cache," in ASPLOS-V, pp. 137-148, Oct. 1992

[6] A. Smith, "Cache Memories," ACM Computing Surveys, 14(3), pp.473-530, 1982

[7] V. Knapp, "Virtually Addressed Caches for Multiprogramming and Multiprocessing Environments," Univ. of Washington, TR85-06-02, June 1985

[8] J. Hennessy and D. Patterson, Computer Organization & Design The Hardware/ Software Interface, Morgan Kaufmann Publishers, 1994

[9] R. Gustafan and F. Shapiro, "IBM 3081 Processor Unit : Design Considerations and Design Process," IBM Journal of Research and Development, 26(1), pp 12-21, 1982

[10] Madhusudhan Talluri, et al., "Tradeoffs in Supporting Two Page Sizes," The 19th Int'l Symp. on Computer Architecture, 20(2), pp.415-424, 1992

[11] G. Taylor, P. Davies, and M. Farmwald, "The TLB Slice-A Low-Cost High Speed Address Translation Mechanism," The 17th Int'l Symp. on Computer Architecture, pp.355-363, 1990

[12] S. Mirapuri, M. Woodacre, and N. Vasseghi, "The MIPS R4000 Processor," IEEE Micro, 12(4), April, 1992

[13] D. Dobberpuhl, "A 200MHz 64b Dual-Issue CMOS Microprocessor," The 39th Int'l Solid-State Circuits Conference, IEEE, pp.106-107, Feb. 1992

[14] J. Pomerene, T. Puzak, et al., "Mechanisms for Acceleration of Cache References," IBM Technical Bulletin, 25(3B), pp.1740-1745, Aug. 1982

[15] K. Hua, A. Hunt, et al., "Early Resolution of Address Translation in Cache Design," IEEE int'l Conference on Computer Design : VLSI Computers & Processors, pp.408-412, Sept. 1990

우 종 정

1982년 경북대학교 전자공학과 학사

1990년 Univ. of Texas at Austin 전기 및 컴퓨터 공학과 석사

1993년 Univ. of Texas at Austin 전기 및 컴퓨터 공학과 박사

1993년~현재 성신여자대학교 조교수

관심분야 : 병렬처리 시스템, 컴퓨터 구조, 정보검색