Low Power TLB Design for High Performance Microprocessors

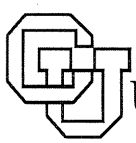
Srilatha Manne
Department of Electrical and Computer Engineering
University of Colorado at Boulder

Artur Klauser
Department of Computer Science
University of Colorado at Boulder

Dirk C. Grunwald
Department of Computer Science
University of Colorado at Boulder

Fabio Somenzi
Department of Electrical and Computer Engineering
University of Colorado at Boulder

CU-CS-834-97



University of Colorado at Boulder
DEPARTMENT OF COMPUTER SCIENCE

Low Power TLB Design for High Performance Microprocessors

Srilatha Manne
Dept. of Electrical and Computer Engineering
University of Colorado at Boulder

Artur Klauser
Dept. of Computer Science
University of Colorado at Boulder

Dirk C. Grunwald
Dept. of Computer Science
University of Colorado at Boulder

Fabio Somenzi
Dept. of Electrical and Computer Engineering
University of Colorado at Boulder

CU-CS-834-97

March 1997

University of Colorado at Boulder

Technical Report CU-CS-834-97
Department of Computer Science
Campus Box 430
University of Colorado
Boulder, Colorado 80309

Srilatha Manne
Dept. of Electrical and Computer Engineering
University of Colorado at Boulder

Artur Klauser
Dept. of Computer Science
University of Colorado at Boulder

Dirk C. Grunwald
Dept. of Computer Science
University of Colorado at Boulder

Fabio Somenzi
Dept. of Electrical and Computer Engineering
University of Colorado at Boulder

Low Power TLB Design for High Performance Microprocessors

March 1997

Abstract

Modern microprocessors consume large quantities of energy, which is detrimental to both the battery lifetime and to the reliability of the processor. Much of the energy consumption comes from the memory hierarchy of the processor. Previous studies [10] have explored cache designs and have shown that certain cache sizes have better power/performance ratios than others. But another component of the memory hierarchy, the translation look-aside buffers or TLBs, has not been studied from the perspective of reducing power dissipation. Most present day TLBs are fully associative, and content addressable memories (CAMs) are used to implement them. This paper proposes a banked associative design for TLBs (BA-TLB), and presents results for the two approaches for both single process and multi-process environments. Banked associative designs consume less power than fully-associative TLBs (FA-TLB) since only half the CAM entries are looked up on each access to the TLB. Experiments show that BA-TLBs perform as well as FA-TLBs. Also, given the same, fixed TLB power budget for BA-TLB and FA-TLB designs, the BA design outperforms the FA design by an average of 83%.

1 Introduction

Energy consumption and power dissipation in microprocessors is of critical importance for many applications. For portable applications, high energy consumption means a shorter battery life. For non-portable applications, power dissipation affects reliability, and in many cases, performance. The performance of some high performance processors is restricted by the power dissipated during operation. Processors like the Alpha 21164 are specified to operate at 600 MHz, although they are structurally capable of operating at frequencies of 767 MHz or higher [1]. But this requires a larger, more expensive package, and hence is cost and or area prohibitive for many applications. Therefore, low power design is no longer just an issue for portable applications, but is of critical importance for all future, high performance processors.

Previous work has shown that most of the energy consumed in a microprocessor is primarily due to two subsystems: Clocks and memory [5]. Gonzalez reported in [5] that 50% to 80% of the power consumption on a chip is due to these two systems, with memory generally accounting for

		Alpha 21064	Alpha 21164	MIPS R10000	AMD 29k	this paper
	Pagesize	8 kB	8 kB	4 kB	8 kB	8 kB
	Inst/Cycle	2	4	4	1	1
	Subblocking	no	no	2	no	no
DTLB	Entries	32	64	64	64 shared	32–128
	Associativity	full	full	full	full	full, 1/2
	Banks	1	1	1	1	1, 2
ITLB	Entries	8	48	8 shared	64 shared	32–128
	Associativity	full	full	full	full	full, 1/2
	Banks	1	1	1	1	1, 2

Table 1: Typical TLB configurations

half of that. The design of the *StrongArm* processor has confirmed these numbers. The clocks on Digital's Alpha 21064 processor, for example, dissipated 65% of the power on chip. The *StrongArm* chip from Digital was able to reduce energy consumption by gating its clock and reducing the load on the clock. Both these schemes reduced power from 0.8 Watts to 0.5 Watts [3].

Work in low power memory hierarchy design has focused primarily on cache design [10], and has shown that although larger caches provide a lower miss rate, they are not always energy efficient due to long bit lines. But this work did not take into consideration the power cost of misses in the cache. A miss would require the processor to go off chip to retrieve data, which is costly both in terms of performance and energy. Another recent trend [4, 10] is to remove the memory hierarchy completely, and instead embed a simple processor inside a large dynamic RAM (DRAM). This is beneficial for both power and performance since large quantities of data are easily accessible by the CPU. But research in this area is very preliminary, and embedded RAM processors will not be replacing high performance processors any time soon.

This paper explores the power/performance cost of another key component of the memory hierarchy system: translation look-aside buffers or TLBs. TLBs are small caches, used to translate from virtual to physical addresses in processors that use virtual memory. The TLB is accessed simultaneously with every access to the data and instruction caches. Both the cache and TLB lookup must be valid in order for the memory operation to complete in one cycle. TLBs are generally very small, but are expensive in terms of power consumption because they are fully associative. The power consumed in a 32 entry CAM with 37 bits per word is equivalent to more than three times the power consumed in a 72 bit adder, for example. Since the TLB is referenced on every access to

the memory subsystem, the power cost can be substantial. Table 1 shows the TLB configurations for various microprocessors.

This paper specifically explores TLB design in a high performance environment. We show TLB simulation results for various sizes and replacement policies. We look at a design alternative to the FA-TLB referred to as a **Banked Associative TLB** (BA-TLB) design. In particular, the BA-TLB reduces the associativity of the TLBs in half while retaining the same size, which reduces the number of entries looked up on each access by 50%. We show that BA-TLBs perform as well as FA-TLBs, and, given the same TLB power budget for FA-TLB and BA-TLB configurations, BA-TLBs significantly outperforms the FA-TLB design.

Section 2 introduces general memory hierarchy issues, and discusses the specifics of CAM design. The experimental method is presented in Section 3, and results are shown in Section 4. Future directions for research are discussed in Section 5, and Section 6 concludes the paper.

2 Memory Organization

Most current high performance general purpose processor architectures support paged virtual memory. Operating systems use paging and virtual addressing as mechanisms to achieve address space protection and flexibility in memory management.

In operating systems supporting multi-processing, each process runs in its own *address space*. The process's address space is its personal view of memory. Multiple processes, each within its own address space, can refer to the same *virtual address* without conflicting with each other in memory. The operating system manages the assignment of *physical addresses* to processes. A virtual address in a process is mapped to a specific physical address in memory by a hardware structure called the *memory management unit*. The same virtual address in another process is mapped to a different physical address in memory. In this way the operating system guarantees that processes will not erroneously or maliciously interfere with each other by changing each others memory state.

2.1 Virtual to Physical Address Translation

The most common solution for mapping virtual to physical addresses is paging. Here, the virtual address space is partitioned into equal sized regions called *pages*. The physical address space is also partitioned into regions of the same size, called *page frames*. To map virtual to physical addresses, the memory management unit uses a *page table* as depicted in Fig. 1. Each virtual address is split

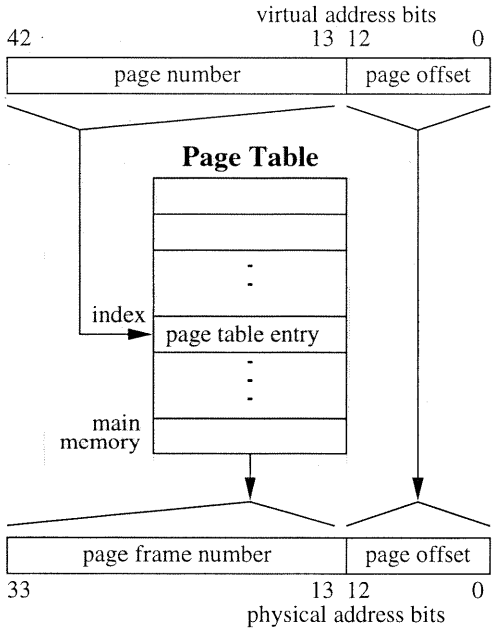


Figure 1: Virtual to physical address mapping. Numbers shown are for the Alpha 21164 processor.

up into two fields, a *page offset* and a *page number*. The page offset is the displacement from the beginning of a page and does not need to be translated. The page number is used to index into the page table and find the *page table entry* (PTE) corresponding to this page. Each PTE is typically 4 to 16 bytes long and contains the page frame number and additional information. The page frame number is then concatenated with the page offset to form the complete physical address (see Fig. 1). The number of bits in the page frame number does not necessarily have to match the number of bits in the page number. Typically, the virtual address space has on the order of 32 to 64 bits and the physical address space has on the order of 30 to 45 bits.

2.2 Translation Lookaside Buffer (TLB)

Using the virtual to physical address translation phase explained above means that every memory operation actually has to performs multiple memory accesses, an access to find the physical address in the page table and the actual access to complete the memory operation. This is not desirable as it would slow down the processor by a factor of 2. To circumvent this problem a *translation lookaside buffer* (TLB) is used (see Fig. 2).

A TLB is a small on-chip cache of recently used virtual to physical address translations. The TLB is addressed by page number and returns the corresponding PTE. TLBs usually have 8 to

128 entries and are typically fully associative. To achieve fast memory accesses, the TLB is accessed in parallel to the on-chip cache, so that the physical address from the TLB is available in the final tag comparison of a physically tagged cache. This results in a 1-cycle memory access time if the address translation is found in the TLB and the data item is found in the on-chip cache.

In order to support concurrent lookups of both instruction fetches and data memory operations in the same cycle, the TLB has to be multi-ported or duplicated for the instruction and data paths. Many architectures employ two separate TLBs, one for instruction references and one for data references.

The instruction TLB (ITLB) is used once every cycle for which instruction fetch addresses need to be translated. Instruction fetch addresses need not be translated if consecutive instructions within the same page are accessed. This optimization is used to avoid ITLB lookups for superscalar multi-way issue architectures such as the Alpha 21164, as well as for processors which use consecutive instruction prefetching such as the AMD 29K.

The data TLB (DTLB) needs as many read ports as there may be concurrent data memory accesses supported per cycle. For scalar architectures this is usually only one read port, whereas for superscalar architectures such as the Alpha 21164 [2] or the MIPS R10000 [6] the DTLB has multiple read ports. Each read port consists of the comparator circuit of a CAM for the TLB tag, as well as a normal read port for the TLB data. This requirement makes it more expensive than normal SRAM read ports.

The miss rate in TLBs tends to be very small since each entry in the TLB refers to one page of memory. On the Alpha 21164, each page is 8 KB. Therefore, a 64 entry data TLB maps 512 KB of data. This does not sound like a lot of memory, but programs generally operate with both spatial and temporal locality, and hence 64 entries are good enough to produce very small miss rates. TLB miss rates are usually much less than 1%, and are an order of magnitude less than cache miss rates. However, if the virtual to physical address translation is not found in the TLB, an expensive lookup in the page table has to be initiated and the TLB has to be updated with this information. When a TLB miss occurs, the current memory operation can not be finished in the same cycle and has to be postponed until the TLB miss is resolved, which incurs a latency on the order of tens to hundreds of cycles. The miss may be handled either by hardware or software. In our simulations, we use software miss handling as in the Alpha.

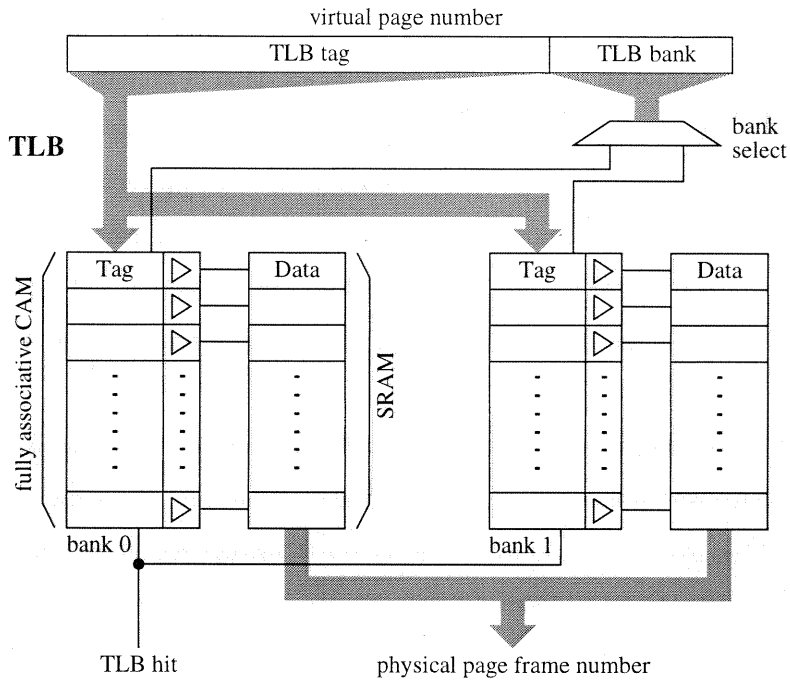


Figure 2: TLB Operation for a banked associative TLB.

2.3 Associative Memory

The tag array for a TLB is built of 9-transistor *content addressable memory* (CAM) cells as depicted in Fig. 3. A CAM cell consists of a standard 6-transistor static RAM cell plus an additional 3-transistors comparator. Two transistors form an XOR gate to compare the stored bit in the cell with the signal present on the bit and $\overline{\text{bit}}$ lines. The third transistor is part of a distributed dynamic NOR gate. All bit cells of a tag share the same match line. All match lines in the CAM are precharged before the CAM lookup. If any of the bit cells detects a mismatch, its NOR transistor will discharge the match line. The CAM is then used to drive the word lines on a SRAM containing the virtual to

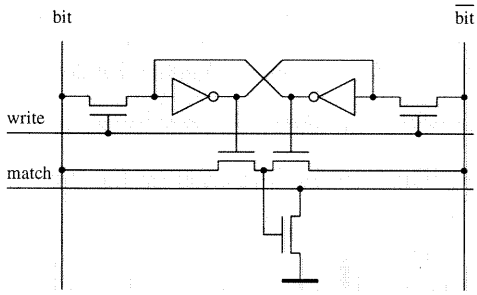


Figure 3: CAM Cell.

physical page translation (Fig. 2). Due to the larger cell size and the requirement of the match lines and bit line drivers, CAMs tend to be fairly large.

TLBs are operated in such a way that only zero or one entry in the CAM can match a virtual address, i.e. all tags in the CAM are distinct. However, in this mode of operation it is guaranteed that at least $n - 1$ match lines are discharged on each access to an n entry TLB. Thus each charge and discharge cycle forces the TLB to dissipate a lot of power because it has to charge and discharge most of its match lines.

The power dissipated by CAMs can be characterized relative to equivalent power consumption for a minimum sized inverter in the same process. Depending on the performance requirements of the CAM, each bit of the CAM dissipates the power equivalent to approximately 0.65 to 1.15 inverters. The faster the CAM, the larger the power dissipation. CAMs also have an intrinsic power dissipation value equivalent to an average of 18 inverters per word. This is the power inherent to the CAM regardless of the size of the core array. Therefore, a 32 entry CAM with 37 bits per word would, on the average, dissipate the same power as $32 \times (37 \times 0.9 + 18) = 1642$ small inverters. The power dissipation of a NAND gate is approximately equal to 2 small inverters. Therefore, the CAM consumes as much power as 821 NAND equivalent gates. Considering that the CAM is active on every access to the memory subsystem, this can add up to a substantial amount of energy.

2.4 Banked Associative TLB (BA-TLB)

Usually, microprocessors use FA-TLB designs even for large TLB sizes of 64 and above. However, it is known for processor caches [8] that associativity above 8-way gives diminishing returns with respect to less associative caches of the same capacity. Although TLBs might not necessarily behave exactly the same, we assume that after a certain point higher associativity does not noticeably contribute to increased performance.

Figure 2 shows our proposed *BA-TLB design*. The TLB is split up into multiple banks, each of which is fully associative. Each bank consists of its own set of CAM cells for the tag array and SRAM cells for the data array. Part of the virtual page number is used to select a bank and the remaining bits of the virtual page number are routed to all banks for tag comparison. Only the selected bank actually drives the tag on its bit lines in the CAM and activates the SRAM word lines and sense amps to read out the data array. All other banks are not referenced. The output of the data arrays of all banks are then multiplexed onto a single result bus containing the physical page

Entries	8	16	32	48	64	96	128
Area	1055	1633	2790	3947	5103	7417	9730

Table 2: TLB area expressed in register bit equivalents (rbe) for 37 tag bits and 32 data bits

frame number. Note that the control signal for this final multiplexer is the same as the bank select signal, so it is already driven very early in the TLB access cycle. We assume that the delay due to the additional bus length and multiplexer are small compared to the access time of one bank. In our simulations we use the least significant bit of the tag to select the bank.

The BA-TLB has some interesting features in terms of power and performance. Assuming that the total number of entries in the TLB is kept constant for BA-TLB and FA-TLB designs, the following observations can be made:

- Due to the fact that only 1 out of n banks is active at any access cycle, the power consumption for the BA-TLB design is roughly $1/n$ of a FA-TLB design.
- The hit rate of a BA-TLB is close to the hit rate of a FA-TLB if each bank still contains a sufficient number of entries.
- Cycle time is usually decreased because bit line wire length is reduced by a factor of n within each bank.
- Chip area for the BA-TLB design is moderately larger because of the need to replicate bit line drivers, sense amps and additional logic for bank selection and output data multiplexing. Chip area estimates for FA-TLBs [11] are shown in Tab. 2.

On the other hand, replicating a FA-TLB with x entries to obtain a n-way BA-TLB where each bank still has x entries has the following properties:

- Power cost stays approximately the same as only one bank is accessed per cycle and the number of entries in this bank is the same as the number of entries in the FA-TLB design.
- TLB performance increases due the the n-fold increase of overall TLB capacity. TLB performance approaches the performance of a fully associative design with n times the number of entries and n times the power cost.
- Cycle time stays approximately the same because each bank of the BA-TLB is just a copy of the original FA-TLB.

- The chip area increases approximately by a factor of n.

3 Experimental Methodology

All experiments presented were run using the ATOM [9] software instrumentation system from Digital Equipment Corporation. ATOM takes instrumentation code written by the user and links it with the executable. For example, we instrumented the SPEC92 and SPEC95 benchmarks with ATOM so that a TLB simulation was performed on each access to an instruction for the ITLB, and on each load or a store operation for the DTLB. Since ATOM only instruments the program currently running, it does not simulate any operating system calls. Therefore, the results produced will be optimistic in their values, but this is not necessarily a concern when comparing the relative performance of different TLB designs.

In particular, the following parameters were varied in the simulations.

- The size of the data TLB (DTLB) and instruction TLB (ITLB).
- Single and multiple process simulation.
- Replacement policies: Least Recently Used (LRU), Round Robin (RR), Random (RAND).

3.1 Basic TLB design

The simulation model for the basic TLB was based on the Alpha 21164 processor. Table 1 gives the relevant information for the processor. The simulated TLB contains the tag associated with a particular page and the process ID so that multi-processing environments could be simulated. The TLB sizes chosen for the simulations were based on existing processor TLB sizes, and on the potential growth in TLB sizes for the future. In general, the only restriction we placed on the sizes was that the ITLB was never bigger than the DTLB, since the DTLB also had the added burden of handling both page misses and page table misses. We simulated a 2-level page handling hierarchy where the page table was also stored in virtual memory. Hence a single page miss could produce multiple misses in the TLB. On the 21164, a page miss could result in another page table miss in the TLB, after which miss handling resorts to physical addressing of the page table. Our simulation model uses the same TLB miss handling strategy. Each page was fixed to be 8 kB in size, and each page in the page table referred to 8 MB since there are 1 k PTEs in each page of the page table.

A page miss on the 21164 is handled by *PALcode*, which is a special processor state used to implement a variety of operating system primitives. Subsequent misses to the TLB are also handled in *PALcode*. The cost of a page miss and a page table miss was constructed in terms of number of *PAL* instructions plus the cost of pipeline flush and drain operations which occur whenever the processor begins executing *PALcode*. Therefore, the cost of a data page miss and a page table miss were approximated to be 34 and 43 instructions respectively. The cost of an instruction page miss was approximated to be 38 instructions. The number of instructions executed in the program was used as an approximation of execution time for switching between processes when simulating a multi-process environment.

In the single process simulations, it was assumed that the TLB would not be corrupted by another intervening process. This is the best case in terms of miss rate. We also simulated the *worst case multi processing* environment where the TLB is thrashed between activations of the same process. This is the worst case miss rate for the TLBs since cold misses will occur every time the process is activated.

3.2 Replacement Policies

LRU will usually produce the best miss rates since this minimizes the possibility of conflict. Unfortunately, the cost of implementing this policy in hardware is prohibitive, and cheaper solutions are generally used. RR replaces entries in strict sequence. Each entry in a n entry TLB is replaced every n-th miss. The problem with RR replacement is that it has no concept of how often an entry is used. Therefore, its performance suffers when there are entries in the TLB that are accessed frequently, but are also replaced frequently due to a large miss rate. RAND replacement is easy to implement, but generally has a higher miss rate than either LRU or RR replacement.

3.3 Benchmarks

We use a number of programs from the SPEC92 and SPEC95 benchmark suites for our simulations. We use both, C and Fortran, integer and floating point intensive programs to cover a wide spectrum of application behaviors.

		DTLB			
	ITLB	32	64	96	128
gcc	shared	3984 / 3989	390.4 / 400.6	132.2 / 146.8	64.12 / 82.49
	16	2862 / 2883	—	—	—
	32	—	249.6 / 289.4	—	—
	64	—	249.7 / 288.9	105.1 / 156.1	66.75 / 134.4
hydro2d	shared	83.04 / 86.99	15.19 / 24.79	1.45 / 16.98	0.02 / 16.91
	16	300.6 / 318.3	—	—	—
	32	—	42.72 / 83.95	—	—
	64	—	42.72 / 83.95	0.06 / 63.46	0.05 / 63.46
spice	shared	5139 / 5144	459.1 / 468.2	102.7 / 115.8	22.15 / 41.73
	16	15024 / 15044	—	—	—
	32	—	1500 / 1534	—	—
	64	—	1500 / 1534	332.2 / 382.5	72.47 / 145.1
ear	shared	0.03 / 4.03	0.03 / 4.03	0.03 / 4.03	0.03 / 4.03
	16	0.11 / 10.19	—	—	—
	32	—	0.11 / 10.19	—	—
	64	—	0.11 / 10.19	0.11 / 10.19	0.11 / 10.19

Table 3: DTLB misses per million references for single / multiple processes using LRU replacement. Empty fields represent cases that were not simulated because the relative ITLB vs. DTLB sizes were assumed to be non-viable.

4 Results

Due to the large variation in absolute miss rates across varying TLB sizes and benchmarks (see Table 3) we present all graphical performance results in relative form with the following general structure:

- The performance of two competing designs is presented relative to each other and normalized to 100. We always build relations such that smaller numbers represent better relative performance.
- Where applicable, graphs show minimum, average, and maximum performance across all benchmarks.
- Each ITLB / DTLB size configuration is represented by two points. One for single process simulations (S) and one for worst case multi process simulations (M). We also simulated 2-process configurations, whose results fall between the single and multiple process results as to be expected. For clarity this data has been omitted from the graphs.

4.1 Replacement Policies

We wanted in insure that any results we achieve with the BA-TLB design would be relevant across different replacement policies. For instance, one replacement policy could conceivably benefit one design over another. Therefore, we simulated results for three different policies: LRU, RR, and RAND replacement.

Figures 4 and 5 show the relative miss rate of RR and RAND compared to LRU, respectively. LRU replacement will almost always produce the best miss rates. We show the minimum, average, and maximum relative differences in miss rates. Note that, as in the case for all data presented, each value is relative based on a fixed TLB configuration. For example, Figure 4 shows that RR replacement for a shared TLB of 64 entries is on the average 75% worse than the same size TLB using LRU replacement. But one cannot then compare RR replacement for a shared TLB of 64 and 128 entries, since these numbers are relative to LRU replacement for 64 and 128 entries, respectively.

One further point to note is that Figures 4 and 5 can be compared to each other for fixed TLB configurations, since both figures are relative to LRU replacement. The data clearly shows that RR replacement is better than RAND replacement, with the exception of S_96 which is an outlier.

4.2 Banked Associative vs. Fully Associative Configuration

There are three major cost metrics in circuit design: performance, area and power. Sometimes improving one metric will also improve another. For example, an area increase may improve cache hit rates, and therefore will generally improve both power and performance because going off-chip to retrieve data is expensive for both criteria. In the case of BA-TLB vs. FA-TLB , the trade-off among the three criteria is conveniently studied from two angles. The first increases the area slightly in BA-TLBs to get a decrease in power dissipation with comparable performance. This is referred to as *fixed performance TLB design*. The second design alternative, referred to as *fixed power TLB design*, keeps power dissipation constant and doubles the area to achieve a significant improvement in performance.

4.2.1 Fixed Performance TLB Design

Figures 6, 7, and 8 show the miss rates for a BA-TLB configuration for LRU, RR, and RAND replacement. Overall TLB sizes are kept the same for each comparison, e.g. a 64 entry FA-TLB is

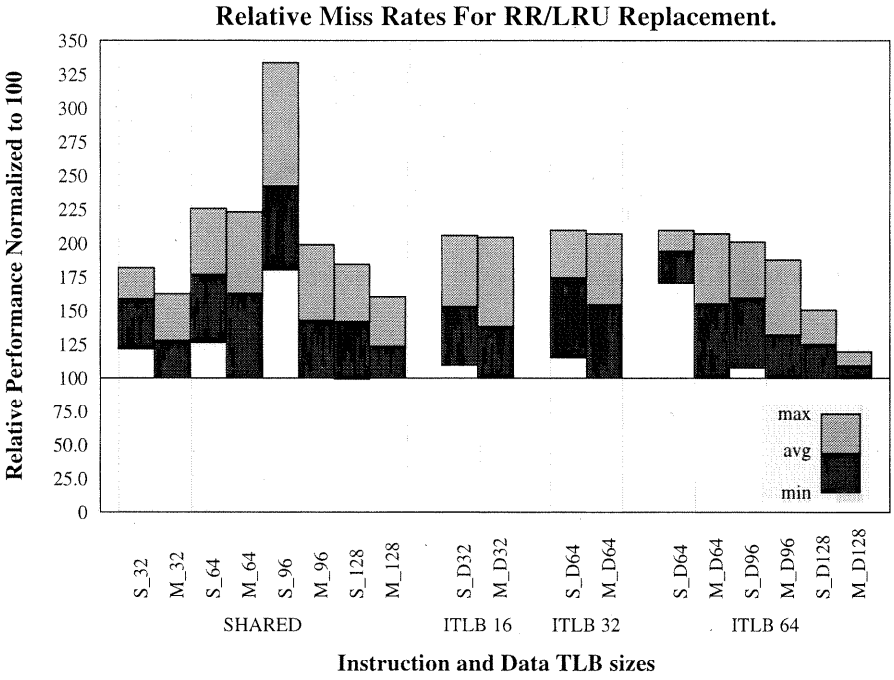


Figure 4: Relative miss rates of Round Robin vs. LRU replacement policies.

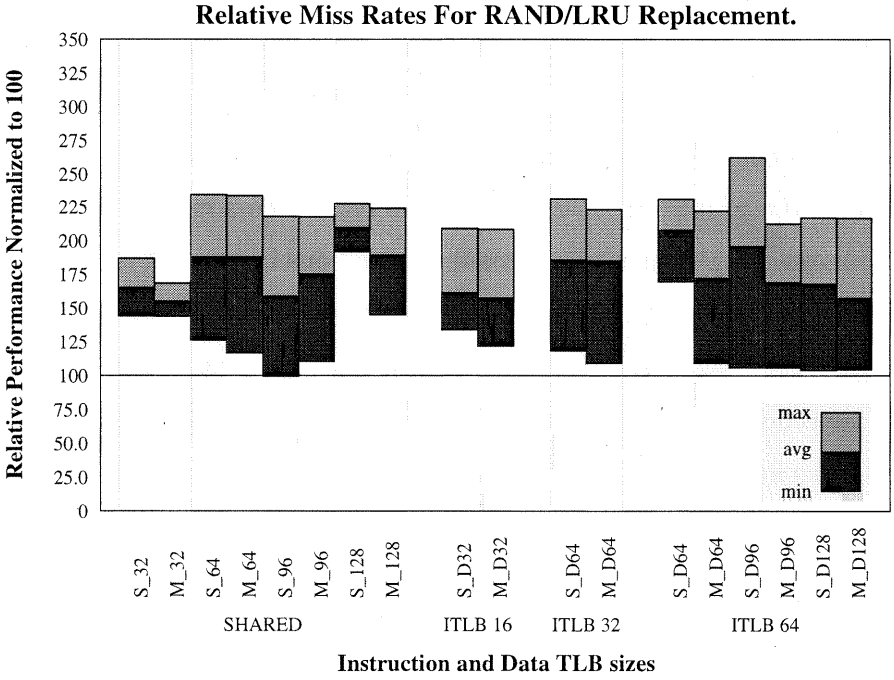


Figure 5: Relative miss rates of Random vs. LRU replacement policies.

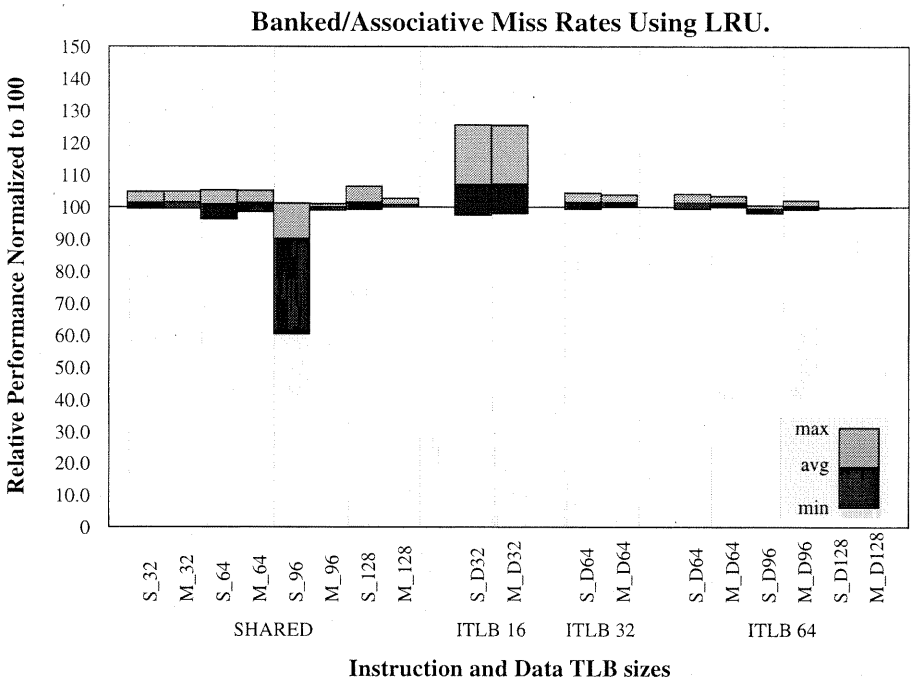


Figure 6: Relative miss rates of BA-TLB vs. FA-TLB designs with equal TLB capacity for LRU replacement policy.

compared against a BA-TLB with 2-banks of 32 entries each. As in all previous figures, smaller is better, and minimum, average, and maximum relative miss rates are displayed. Ideally, the relative miss rates for BA-TLBs would average out around the 100 mark, indicating that BA-TLB, and FA-TLB designs have similar miss rates. In reality the numbers are not far off from optimal. For LRU and RAND replacement, we see an increase of less than 1% in relative miss rates over all TLB configurations, all examples, and single and worst case multi-process simulations. RR replacement even performs marginally better with a BA-TLB configuration. The figures show that the penalty for decreasing associativity is minimal. Also, even this small difference may be further reduced through alternative bank selection schemes that distributes usage of the banks more evenly. One interesting point to note is that the S_96 BA-TLB always performs better than the FA-TLB. This is a case of interference between the particular application and our bank selection scheme that results in resonance.

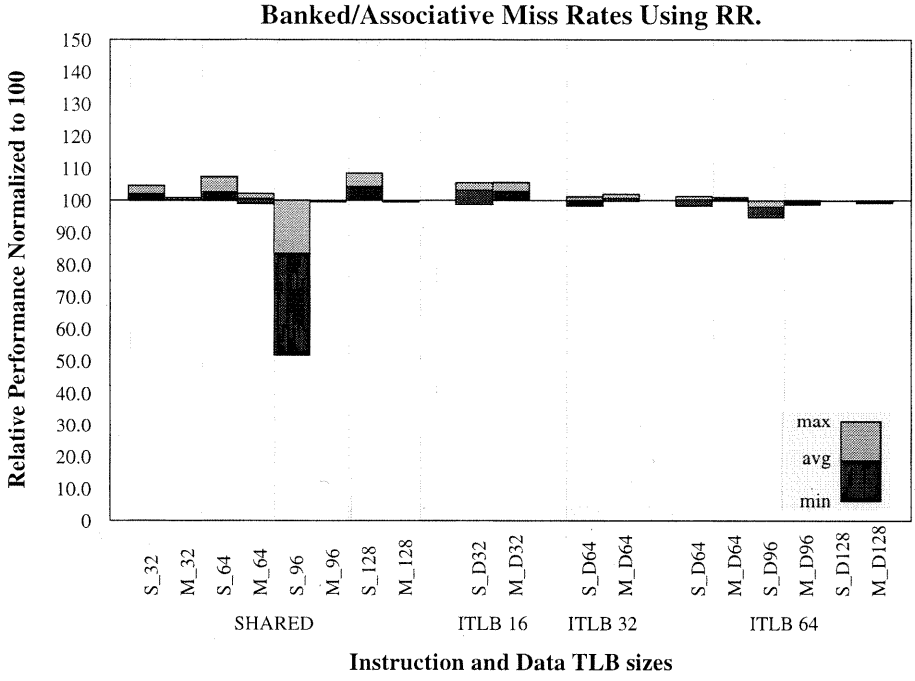


Figure 7: Relative miss rates of BA-TLB vs. FA-TLB designs with equal TLB capacity for Round Robin replacement policy.

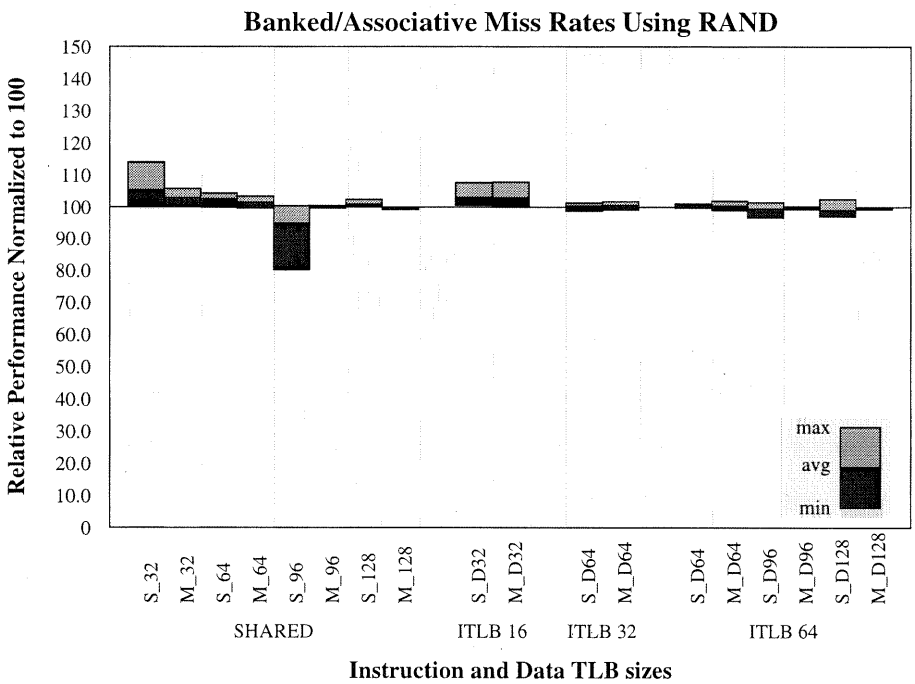


Figure 8: Relative miss rates of BA-TLB vs. FA-TLB designs with equal TLB capacity for Random replacement policy.

4.2.2 Fixed Power TLB Design

The results shown in Figures 6, 7, and 8 are for the cases where the banked system has the same capacity as a fully associative system. Hence, for a small penalty in area due to duplication of peripheral circuitry, the BA-TLB design produces comparable performance with approximately half the power consumption. On the other hand, if area is not an important constraint, and one wants to achieve high performance given a fixed power budget, then for approximately the same power budget as a FA-TLB of n DTLB entries and m ITLB entries, one can use this BA-TLB configuration of $2 \times n$ DTLB entries and $2 \times m$ ITLB entries. The power dissipated in a BA-TLB configuration with twice the capacity is not exactly the same as the power dissipation in the FA-TLB design. The BA-TLB configuration will have higher power dissipation due to longer wires. But, as shown in Figure 9, the miss rate for the BA-TLB implementation given a fixed power budget is significantly better than that of the FA-TLB implementation. Therefore, any extra power consumed by the BA-TLB configuration is more than offset by the power saved due to a significantly smaller miss rate.

Figures 9 shows the relative miss rates for *GCC* between FA-TLB and BA-TLB designs given a fixed power budget. Therefore, the capacity of the BA-TLB is twice that of the FA-TLB design. Data is shown for various TLB configurations. The TLB sizes given are for the size of the FA-TLB. Hence the data for *D32_I16* shows the relative performance of a FA-TLB with 32 entry DTLB and 16 entry ITLB when compared to a BA-TLB with 64 entry DTLBs and 32 entry ITLBs. Data is shown for all 3 replacement policies.

5 Future Research

So far, we have presented work which shows that banking of TLBs is a viable option for processors seeking high performance without paying a large penalty in power dissipation. But there are many other TLB designs and options available which we have not yet investigated. Therefore, we shall present a qualitative look at other TLB designs and discuss the impact these designs would have on power consumption and performance.

The most obvious design change to the TLB modeled in the previous section is the use of *superpages*. Superpages have been proposed as a way to reduce TLB misses without the necessity of increasing TLB size. [11] and [12] discuss the advantages and disadvantages of this design option. Superpages work well when the data being used is accessed in large chunks. On the other

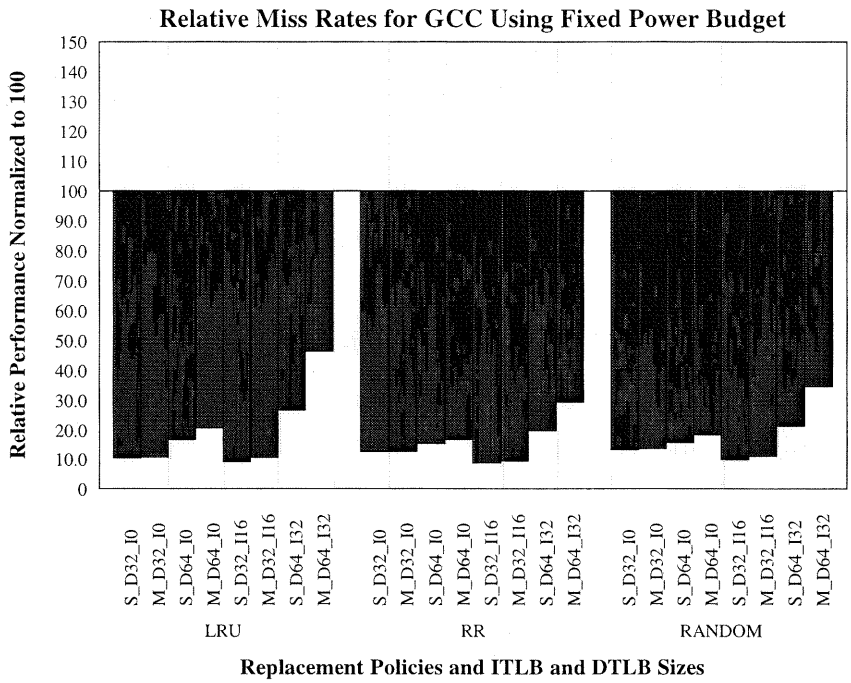


Figure 9: Relative miss rates of BA-TLB vs. FA-TLB designs with equal TLB power budget for the gcc benchmark.

hand, they have the potential for increased internal fragmentation in memory if the data is not accessed consecutively. There is also an extra performance and power cost associated with bringing in large pages from disk to main memory. The obvious solution to this problem is the ability to use multiple page sizes. Most hardware, such as the Alpha processors, support this option. Unfortunately, most operating systems only support multiple page sizes for the kernel, but not for user processes. Therefore, non-kernel processes are fixed to a single page size which is generally in the range of 4 kB to 32 kB. Although we only show banked and non-banked results for a 8 kB page size, we believe that the miss data should scale accordingly for large page sizes. The case could also be made that large page sizes would require a smaller TLB, and hence banking is no longer needed to reduce power consumption in a TLB. But large page sizes and banking, together, could produce better results than either alone, especially for processes with large address space.

Another possible design change to the TLB involves sub-blocking [12]. Sub-blocking in TLBs is similar to sub-blocking in caches in that a single tag is used to access multiple locations in the TLB. Sub-blocking has been presented as a possible replacement for *superpages*. Sub-blocking is an interesting option for reducing power consumption since a single CAM entry can be used to

access multiple entries in the TLB. Hence, using sub-blocking of 2 can cut in half the number of entries in the CAM. The problem is that the performance of the sub-blocked TLB suffers when the pages being used are not in contiguous locations in memory. This is not the case with the BA-TLB model we have proposed. Hence our model is less sensitive to data locality issues. It was shown in [7] that TLB misses can significantly affect the performance of large CAD programs, and these programs do not necessarily benefit from sub-blocked TLBs.

Finally, the data in the previous section has shown that the BA-TLBs performs as well as a fully associative TLB. We think that the performance of the banked associative TLB design can be further enhanced with a better hashing scheme to determine the bank in which the page is to be located. The entries in opposite banks can never overwrite one another, and this can be used to our advantage to further reduce the miss rate. Also, this poses the question of whether there should have been more than 2 banks in the banked design. This was our initial choice to determine the validity of the idea. [8] shows that cache miss rates improve the most when going from direct mapped to a 2-way set-associative design. Increasing associativity further results in decreasing additional gains. Considering our results, there is the possibility that more banking can further reduce power consumption with marginal performance loss.

6 Conclusion

The goal of this paper was to propose a TLB design that dissipates less power than a fully associative TLB while retaining performance. The proposed banked associative TLB does just that. The BA-TLB consumes approximately half the power of the FA-TLB design since each bank is half the size of a FA-TLB and only one bank is active for each TLB access. We have shown that the performance of the BA-TLB is comparable to a FA-TLB for various replacement policies. Also, given a fixed power budget, the BA-TLB produces significantly better performance than a FA-TLB configuration. In the future, we plan to explore further banking options along with other TLB configurations such as full and partially sub-blocked TLBs.

References

[1] Digital Shows Chilled 767 MHz Alpha. *Microprocessor Report*, December 1996.

[2] Dileep P. Bahnadarkar. *Alpha Implementations and Architectures*. Digital Press, 1996.

[3] Dan Dobberpuhl. The Design of a High Performance Low Power Microprocessor. In *International Symposium on Low Power Electronics and Design*, pages 11–16, August 1996.

[4] Richard Fromm, Stylianos Perissakis, Neal Cardwell, Christoforos Kozyrakis, Bruce McGaughy, and David Patterson. The Energy Efficiency of IRAM Architectures. Technical report, University of California at Berkeley, Department of Computer Science, November 1996. To be published in ISCA97.

[5] Ricardo Gonzalez and Mark Horowitz. Energy Dissipation in General Purpose Microprocessors. *IEEE Journal of Solid-State Circuits*, 31(9):1277–1284, September 1996.

[6] Joe Heinrich. *MIPS R10000 Microprocessor User's Manual*. MIPS Technologies, 1995.

[7] Srilatha Manne, Dirk Grunwald, and Fabio Somenzi. Rememberance of Things Past: Locality and Memory in BDDs. In *To appear in Design Automation Conference*, August 1997.

[8] Steven A. Przybylski. *Cache and Memory Hierarchy Design*. Morgan Kaufmann, 1990.

[9] Amitabh Srivastava and Alan Eustace. ATOM: A System for Building Customized Program Analysis Tools. *SIGPLAN Notices*, 29(6):196–205, June 1994.

[10] C-L. Su and A.M. Despain. Cache Design Trade-Offs for Power and Performance Optimization: A Case Study. In *International Symposium on Low Power and Design*, pages 63–68. IEEE, 1995.

[11] Madhusudhan Talluri and Mark D. Hill. Surpassing the TLB Performance of Superpages with Less Operating System Support. In *International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 171–182, October 1994.

[12] Madhusudhan Talluri, Mark D. Hill, and Yousef A Khalidi. A New Page Table for 64-bit Address Spaces. In *Symposium on Operating Systems Principles*, pages 184–200, 1995.