IEICE TRANS. INF. & SYST., VOL. E80-D, NO. 7 JULY 1997

LETTER

SEWD: A Cache Architecture to Speed up the Misaligned Instruction Prefetch

Joon-Seo YIM†, In-Cheol PARK†, *and* Chong-Min KYUNG†, *Nonmembers*

SUMMARY In microprocessors, reducing the cache access delay and the number of pipeline stall is critical to improve the system performance. In this paper, we propose a *Separated Word-line Decoding (SEWD)* cache to overcome the pipeline stall caused by the misaligned multi-words data or instruction prefetches which are placed over two cache lines. SEWD cache makes it possible to perform misaligned prefetch as well as aligned prefetch in one clock cycle. This feature is invaluable because the branch target addresses are very often misaligned (Percentage of misalignment in the cache is 8 to 13% for 16-byte caches). 8 Kbyte SEWD cache chip was implemented in 0.8 μm DLM CMOS process. It consists of 489,000 transistors on a die size of $0.853 \times 0.827\,\text{cm}^2$.

key words: *cache, microprocessor, pipeline*

1. Introduction

The advancement of VLSI implementation technology leads to high performance microprocessors that operate above 300 MHz with more than three million transistors on a single chip. The performance of cache which occupies up to 40% of whole chip area has become a bottleneck in the state-of-the-art microprocessors. Hence, many researches are now focusing on the performance enhancement of cache in microprocessors. In the architecture level, the optimal cache size, the replacement algorithm and cache miss handling schemes have been studied for a long time [1], [2]. Various techniques were investigated to speed up the data transfer between cache and processor [3]. To this end, such performance-improving techniques as write buffering, reordering, and line buffering are practically employed to minimize the pipeline stall in the microprocessors [4]. In the circuit level, techniques such as pipelined decoding [5], wave-pipelining [6], and self-timing [7] are actively being studied.

Most of the above researches are focused on minimizing the cache access delay regardless of architectures and memory access patterns of microprocessors. In any microprocessor adopting the cache architectures developed so far, the pipeline should stall at least two clock cycles until the required full words data are available if the operands are placed over two cache lines. Therefore, a new cache architecture which can effectively handle such a misaligned data access is required to enhance the microprocessor performance by removing the pipeline stalls.

For an example shown in Fig. 1, the misaligned data access needs two extra two cycle penalties. In the first clock cycle, lower line ($line[i]$) is read out from cache and saved in a read buffer temporally. Upper line ($line[i+1]$) is read out in the second cycle. Then, the lower word of read buffer is aligned with the upper word using a load aligner which is shown in Fig. 2. During these two extra cycles, the pipeline is stalled, which is called a *cache line boundary problem* due to the address misalignment.

In superscalar microprocessors, the bandwidth of instruction prefetch should be large in order that there is no shortage of instructions to execute in multiple pipes. If the cache access is aligned in one line, the number of instruction prefetched at a time is the same

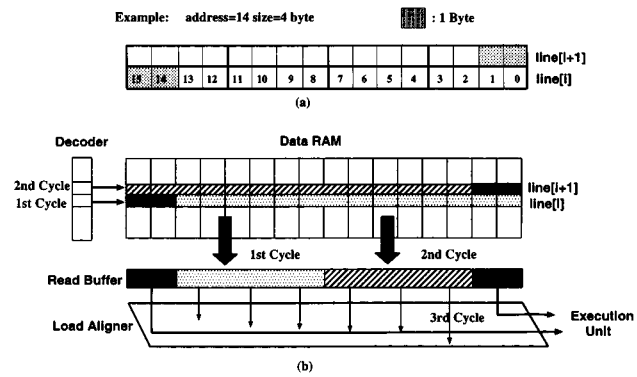

Fig. 1 (a) Misaligned data across the two cache lines and (b) its handling needs extra cycle penalties.

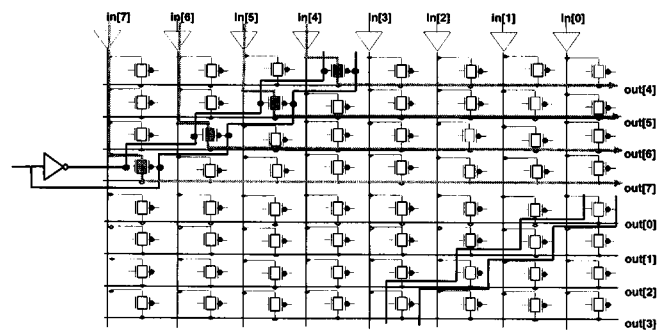

Fig. 2 One byte cell of load aligner: 4 bit shifting case.

Manuscript received March 12, 1997.

†The authors are with the Department of Electrical Engineering, Korea Advanced Institute of Science and Technology 373-1 Kusong-dong, Yusong-gu, Taejon 305-701 Korea.

as the cache line size. In CISC microprocessors, especially in VAX or x86, instruction length varies from 1 byte to 15 bytes [8]. Therefore, the branch target positions can randomly placed anywhere within the cache line. Moreover, the frequency of branch instruction is so high, one per four or five instructions [8], that the branch target address is very often misaligned. To minimize the address misalignment penalty, the high level compilers often fill NOP (No Operation) instructions into an original program to align the target branch to the cache line boundary. According to the benchmark reports, the aligned program improves the performance by 30% over the misaligned program [9].

In this paper, we propose a "*Separated Word-line Decoding (SEWD)*" architecture which can access the misaligned branch target instruction or the misaligned data in one cycle as well as aligned instruction/data. Moreover, SEWD architecture improves the cache hit detection timing which is a critical path in cache circuits.

This paper is organized as follows. Section 2 describes the SEWD architecture. Section 3 shows the chip implementation. The trace-driven simulation result for the address misalignment is shown in Sect. 4.

2. SEWD Architecture

To access the two adjacent cache lines simultaneously, we need a modification in the data RAM architecture as shown in Fig. 3. RAM array is divided into two planes and the decoder is positioned between the two RAM planes. This bipartition of the RAM plane is also helpful for reducing the word line access time.

Two adjacent cache lines are not generally in successive position in the main memory. To handle the misalignment in a single cycle, therefore, we should know whether the two adjacent cache lines have the same tag address or not. That is, we read $tag[i]$ and $tag[i+1]$ and compare them with physical address simultaneously to give $hit[i]$ and $hit[i+1]$. To make this possible, the adjacent tag RAM lines should also be accessed simultaneously. But this is impossible in a traditional tag RAM architecture shown in Fig. 4 (a).

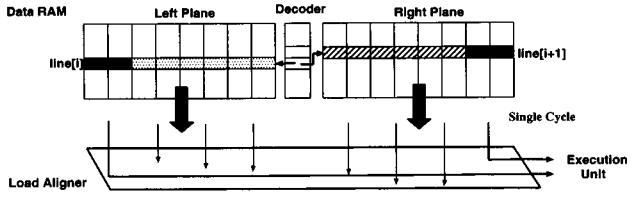

Fig. 3 Modification of architecture to access the adjacent two cache lines simultaneously.

To resolve the problem, we divide tag RAM into even and odd parts as shown in Fig. 4 (b). The LSB of line index in the address (odd/even bit in Fig. 4 (b)) determines whether the even or odd array is accessed in the SEWD architecture shown in Fig. 4 (b). The result of simultaneous tag read can be one of three cases: complete hit (both $line[i]$ and $line[i+1]$ are `hits`), partial hit (one of them is `hit`), and complete miss (both are `misses`). For the case of complete hit, the two adjacent cache lines can be accessed in one cycle. Otherwise, cache miss handling process is started. If $line[i]$ is hit and $line[i+1]$ is miss, the $line[i]$ is forwarded to the execution unit during the cache miss handling for the $line[i+1]$. If the $line[i]$ is miss, the pipeline has to be stalled waiting for the $line[i]$. As the cache hit ratio is usually over 98% [8], most of the misaligned instruction/data can be accessed in one cycle.

In a traditional architecture, the decoder circuits are the same for the left and right plane as shown in Fig. 4 (a), which means that the two adjacent word-lines can not be driven simultaneously. We modified the decoder circuit to make the adjacent cache lines be accessed concurrently. For a misaligned address, $line[i]$ in the left plane and $line[i+1]$ in the right plane are activated. Otherwise, $line[i]$ is driven both for the left and right plane. That is, in the left plane, $line[i]$ is selected regardless of misalignment or not, but in the right plane, $line[i]$ is selected for the aligned case and $line[i+1]$ for the misaligned case. To make this mechanism possible, the left plane uses a normal decoder as shown in Fig. 5 (a), while the right plane uses a SEWD decoder as shown in Fig. 5 (b). If an address is not aligned, `side` signal is activated to drive $line[i+1]$, otherwise `self` signal is activated to drive $line[i]$ in a SEWD decoder. SEWD decoder is also used for the odd part array of tag RAM.

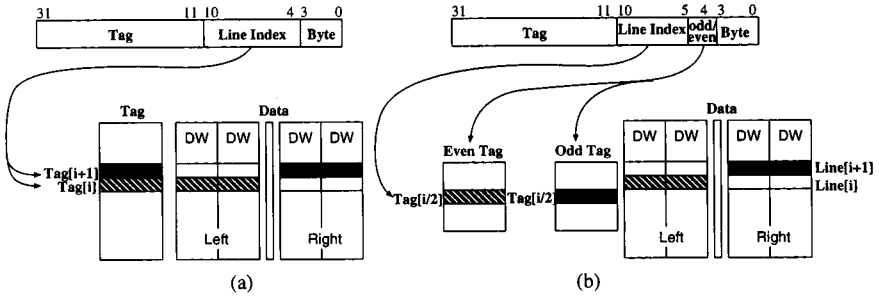

Fig. 4 (a) Traditional architecture and (b) SEWD architecture.

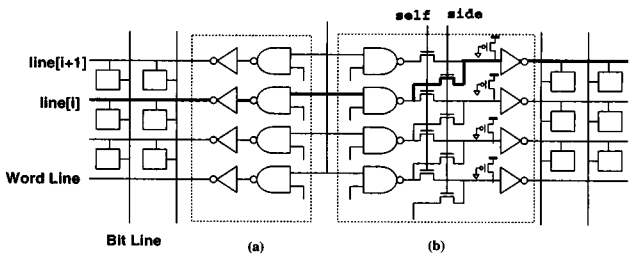

Fig. 5 (a) Normal decoder at left plane and (b) SEWD decoder at right plane.

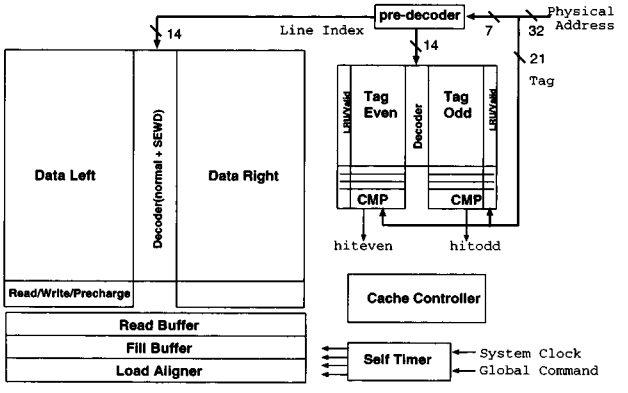

Fig. 6 Block diagram of SEWD cache.

3. Test Chip Design

We have designed a 4-way set-associative 8 kbyte on-chip cache to verify the proposed SEWD architecture. As shown in Fig. 6, the fabricated test chip includes data RAM which consists of 128 lines, where each line consists of 16 bytes. Among the 32-bit physical address, upper 21 bits are used for the tag, middle 7 bits are for the line index, and lower 4 bits select the byte. Pseudo LRU (Least Recently Used) algorithm [8] is used for the cache line replacement scheme. Cache controller supports SEWD cache miss handling.

Tag array is broken into two parts to implement SEWD architecture, therefore, it needs extra read/write circuits and comparators. For the test chip, the extra area overhead due to the SEWD architecture is only 2.35% as shown in Table 1 (Data RAM area is the same in both cases, as in conventional cache, the data RAM area is bipartitioned to speed up the cache line access anyway). Furthermore, as SEWD tag reduces the bit-line length to half, the reduced bit-line capacitance improves the hit detection time from 7.18 ns of conventional architecture to 5.59 ns a saving by 22%, which is another important benefit of the SEWD architecture.

In the test chip, to perform the complicated micro-level operations within one clock cycle and to minimize the power consumption, a self-timed circuit technique was used such as a pulsed predecoder and a strobed sense amplifier [7].

Figure 7 shows the photograph of the test chip,

Table 1 Area comparison between conventional and SEWD architecture.

Block	Conventioanl	SEWD
Tag RAM	3.55 mm^2	4.11 mm^2
DATA RAM	20.7664 mm^2	20.7763 mm^2
Total	24.3164 mm^2	24.8863 mm^2
Ratio		+2.35%

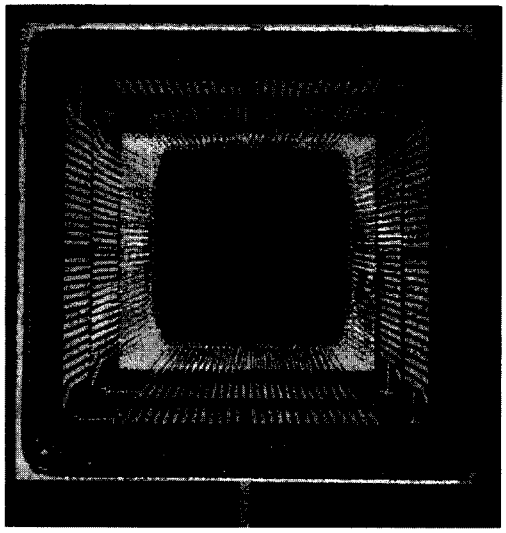

Fig. 7 Photograph of SEWD test chip.

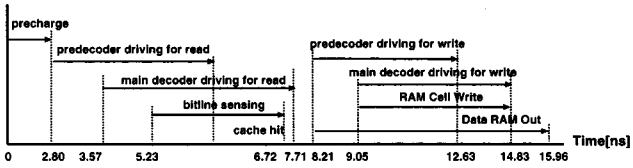

Fig. 8 Various operation of the SEWD cache occurring during one clock cycle.

which consists of 77,184 bit RAM cells based on 6 transistor SRAM cell having the size of 11.2 μm $\times$ 18.8 μm. The total number of transistors is 489,000 and the chip area is 0.853 $\times$ 0.827 cm^2. The chip was proven to work correctly at 60 MHz with the timing budget shown in Fig. 8.

4. Trace Simulation

To see how often the memory addresses are misaligned, the trace-driven simulation is applied for benchmark programs [10]. We also experimented with x86 application such as Windows 3.1 on the x86 instruction set simulator. The result is shown in Table 2 for 8 kbyte cache size. For the line size of 16 bytes, which is very usual in most microprocessors, 7.9% to 13.7% of accesses were found as misaligned. It manifests the significance of SEWD architecture in the instruction cache.

5. Conclusion

In this paper, we propose a separated word-line decod-

Table 2 Percentage of the misaligned case for the instruction prefetch for various benchmark programs.

Line Size (Bytes)	008 espresso	022 li	023 eqntott	026 compress	windows boot
8	20.318	19.906	18.697	28.382	17.399
16	9.582	10.903	12.469	13.732	7.928
32	4.217	6.535	8.525	7.331	3.787
64	2.334	3.975	2.147	4.161	2.006

ing (SEWD) cache architecture to solve the cache line boundary problem. It reduces the clock cycle penalty due to the address misalignment access, which occurs quite often (up to 20%) for the misaligned branch target instruction prefetch. According to our experiment with trace-driven simulation, about 10% of instruction prefetches are misaligned for the 16-byte cache line size.

In 0.8 μm DLM CMOS technology, the critical path timing was reduced from 7.18 ns to 5.59 ns, at the area overhead of only 2.3%, which is a side, yet important benefit of the SEWD architecture since the cache normally forms the critical path in most high-performance microprocessors.

Acknowledgement

The authors are grateful to Hee-Choul Lee, Tae-Hoon Kim, Bong-Il Park, Chang-Jae Park, who are involved in the design of cache for the K486 microprocessor. This work may be impossible without their endeavor.

References

[1] S. Przybylski, M. Horowitz, and J. Hennessy, "Performance tradeoffs in cache design," Proc. 15th Int. Symp. on Computer Arch., June 1988.

[2] A.J. Smith, "Cache memory," Computing Surveys, vol.14, no.3, pp.473–530, Sept. 1982.

[3] M. Motomura, T. Inoue, H. Yamada, and A. Konagaya, "Cache-processor coupling: A fast and wide on-chip data cache design," IEEE J. Solid-State Circuits, vol.30, no.4, pp.375–382, April 1995.

[4] 82395DX 386^{TM} Smart Cache, Intel Corp., 1990, Order Number 290382-001.

[5] Y. Unekawa, T. Kobayashi, T. Shirotori, Y. Fujimoto, T. Shimazawa, K. Nogami, T. Nakao, K. Sawada, M. Matsui, T. Sakurai, M.K. Tang, and W.A. Huffman, "A 110-MHz/1-Mb synchronous tag RAM," IEEE J. Solid-State Circuits, vol.29, no.4, pp.403–410, April 1994.

[6] K. Ishibashi, K. Komiyaji, H. Toyoshima, M. Minami, N. Ohki, H. Ishida, T. Yamanaka, T. Nagano, and T. Nishida, "A 300-MHz 4-Mb wave-pipeline CMOS SRAM using a multiphase PLL," IEEE J. Solid-State Circuits, vol.30, no.11, pp.1189–1195, Nov. 1995.

[7] A.L. Silburt, R.S. Phillips, G.F. Randall Gibson, S.W. Wood, A.G. Bluschke, J.S. Fujimoto, S.P. Kornachuk, B. Nadean-Dostie, R.K. Verma, and P.M. Diedrich, "A 180 MHz 0.8 μm BiCMOS modular memory family of DRAM and multiport SRAM," IEEE J. Solid-State Circuits, vol.28, no.3, pp.222–232, March 1993.

[8] J.L. Hennessy and D.A. Patterson, "Computer Architecture: A Quantitative Approach," Morgan Kaufmann Publishers, 1990.

[9] T.R. Halfhill, "Intel launches rocket in a socket," Byte, pp.92–108, May 1993.

[10] "SPEC Integer Benchmark Suite: CINT92," 1992.