Proc. 2nd Annual National Symposium on Computational Science and Engineering, Bangkok, 1998, pp.244-248.

A VLSI Design of a Load / Store Unit for a RISC Processor

Primas Taechashong, Prabhas Chongstitvatana
Department of Computer Engineering
Chulalongkorn University
Phaya Thai Road, Bangkok 10330, Thailand
primas@navol.cp.eng.chula.ac.th.

Abstract

This work presents a VLSI (Very Large Scale Integrated circuit) design of a Load/Store unit. It is a cache controller for a RISC (Reduced Instruction Set Computer) processor with superscalar technology. The emphasis is on data cache. The controller has a pipeline structure and can execute MIPS R4000 instructions [1]. There are many approaches to cache design which emphasise the effect of changing design parameters on performance [2]. This work contributes an additional study of the effect of buffer size on performance of a data cache which traditionally has been neglected because it was considered an implementation detail. Two parameters are considered : buffer width and buffer depth. The design has been simulated and verified for correct functionality with respect to a large set of random instructions. To measure the performance, traces on a number of benchmark programs on MIPS R4000 processor were analysed.

1. Introduction

The gap between the speed of processor and memory has widen significantly in the past years and will continue to be so. Cache memory is used to fill this gap [3]. The benefit of cache memory relies on its performance parameters, i.e. cache size vs hit ratio, and must be weighted against the cost. Further improvement of the performance of cache memory can be done by reducing the number of stall, for example, the stall from load miss. The pipeline should be able to continue its operation without waiting as long as execution of that instruction doesn't have data hazard [5]. For store hit, the value in cache doesn't have to be written immediately to enable the processor to continue its operation. The store miss can be managed in the similar fashion to reduce the number of clock waiting for writing to memory. The significant reduction of the number of stall from store hit and store miss can be achieved by using buffers. For example, the design of DEC Alpha AXP processor [4] has included a buffer of size 4 words. The question about the optimal size of this buffer remains open.

This work contributes an additional study of the effect of buffer size on performance of a data cache which traditionally has been neglected because it was considered an implementation detail. The next section presents the working details of the Load Store Unit (LSU). Section 3 describes the method to carry out the experiment which included : verification, generation of address traces and performance evaluation. The last section concludes this study.

2. Load / Store Unit (LSU)

LSU performs the execution of load store instruction which communicate with memory. It is organised as 3 stage pipeline as follows (the processor is a 5 stage piped unit) :

1. R stage (read register file and instruction decoding) decodes an instruction and signal the next stage.
2. X stage (execution) calculates the virtual effective address
3. C stage (cache read) gets data from Dcache and checks cache hit or miss. C stage has 4 possible states :
   - Load hit : data from load can be written to register
   - Load miss : start LMC unit (Load Miss Control). The pipeline continues without stall waiting for data.
   - Store hit : start SHC unit (Store Hit Control) which has internal buffer to store data that is waiting to be written to Dcache. The pipe line continues its operation.
   - Store miss : start SMC unit (Store Miss Control) which has internal FIFO. The pipeline continues its operation.

The organisation of data in SMC FIFO is either Word 32 bits or Double word which can have 32 or 64 bits, the type depends on the type of data (bus interface is 64 bits). The stall can occur in two units : SHC and SMC. SHC stall occurs when SHC buffer is full (by the write from C stage) or E stage refers to the data in cache that has the same address as in the buffer. SMC stall occurs when FIFO is full (by the write from C stage).

3. Experiment

The design is done using a hardware description language Verilog. The verification of the design is done by random instruction testing (similar to Pentium processor [3]). A set of random instruction is generated and is used to test LSU. The value of output vector from the write operations to registers, Dcache and memory, is compared with the value of output vector acquired from the same instructions running in a simulator written in a high level programming language that simulates the operation of LSU. This simulator is carefully check for its correctness. It is not too difficult to do so because its only concerns with the "result" of the execution of instructions and not the timing.

The traces that are used to measure the performance of the design are generated from 2 benchmarks (Table 1).

Benchmark	Instructions	Memory references	Loads	Stores
gzip	117304	34649 (29.54%)	19393 (16.53%)	15256 (13.01%)
diff	8936	3519 (39.38%)	2599 (29.08%)	920 (10.30%)

Table 1 Characteristics of address traces

The test is performed by varying these parameters :

1. Cache size : 1K, 2K, 4K ,8K and 16Kbytes
2. Organization : direct-map and 2-way set association with random replacement policy
3. Buffer size of SHC : 1, 2, 4 and 8 buffers.
4. FIFO size of SMC : 2, 4, 8 words and doublewords.

4. Result

4. 1 Varying cache size and degree of set associativity

Figure 1 shows the improvement of performance when the cache size is increased. The 2-way association is better than a direct-map.

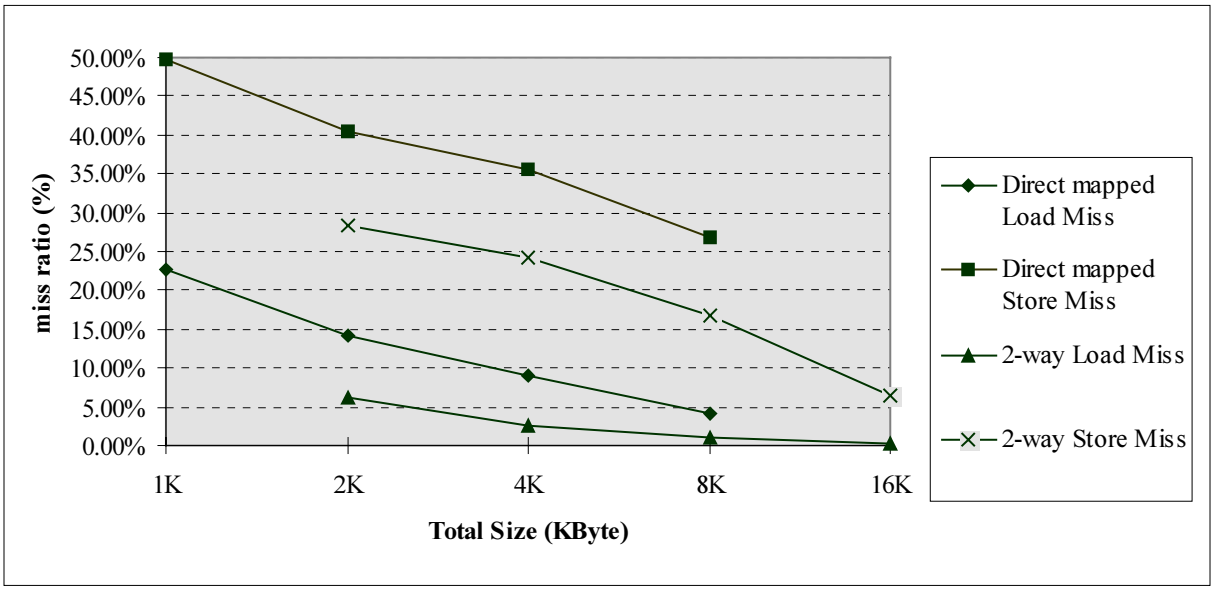

Figure 1: Data cache load and store miss rate as cache size is varied from 1K, 2K, 4K ,8K and 16Kbytes with direct-mapped and 2-way set associative

4.2 Varying FIFO size in SMC

Figure 2 and Figure 3 show that the increase in FIFO depth reduces the number of stall. The type doubleword is better than word. The change in buffer size in SMC does not affect the number of stall in SMC.

4.3 Varying buffer size in SHC

Figure 4 shows that the increase in the number of buffer in SHC reduces the number of stall. Without any buffer, the stall occurs every time the store hit occurs. Buffer size 1 reduces the number of stall to ≅ 45 % and size 2 reduces further 6 %. The size larger than 2 does not reduce the number of stall. This is because those stalls are not the result of buffer full. They are likely to cause by ≅ 39 % the E stage refers to the cache with the same address as in the buffer.

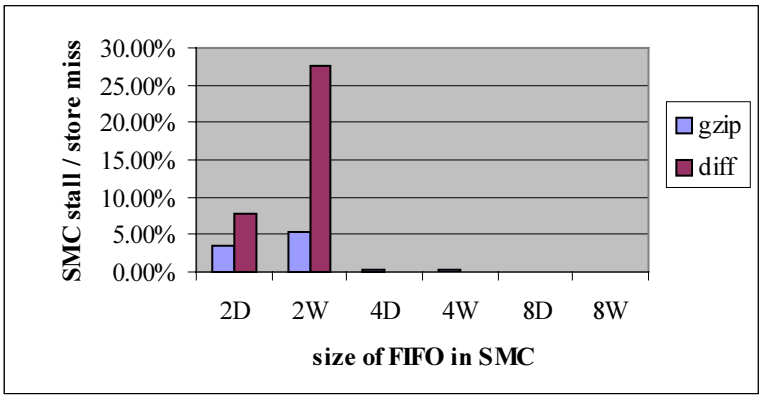

Figure 2: SMC stall as FIFO size is varied from 2, 4, 8 doublewords and 2, 4, 8 words with 1 buffer of SHC

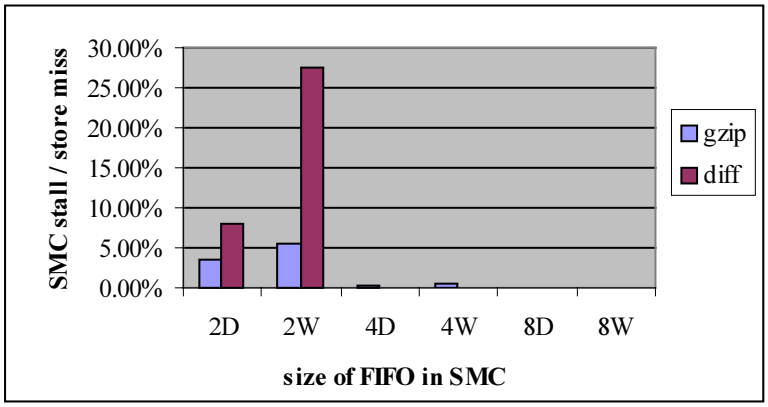

Figure 3: SMC stall as FIFO size is varied from 2, 4, 8 doublewords and 2, 4, 8 words with 2 buffers of SHC

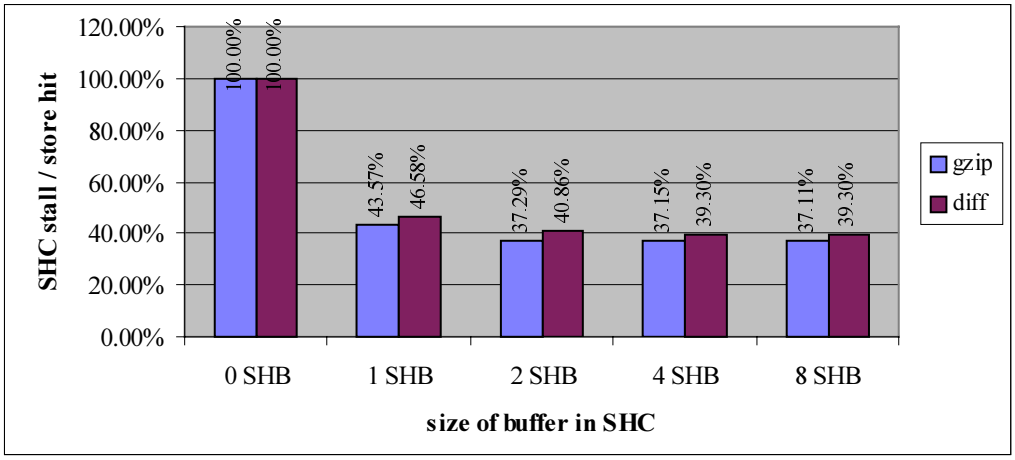

Figure 4: SHC stall as buffer in SHC is varied from 1, 2, 4, 8 buffers with 2 doublewords FIFO in SMC

5. Conclusion

This work presents a design and analysis of a Load/Store Unit which is a cache controller for a RISC processor with superscalar technology. The performance measurement using the trace from two benchmark programs with approximately 100K and 10K instructions. We measured the miss ratio of varying cache size and degree of associativity. To reduce the number of stall by store instructions which write to Dcache or memory, a buffer in SHC and a FIFO in SMC are introduced. We varied the number of buffer in SHC. The result showed that for the size more than two, the number of stall is almost constant which may be caused by the E stage. We varied the depth and the width of FIFO in SMC. The result showed that Doubleword is better than Word and the depth more than 4 reduces the number of stall almost to zero. The future work will concentrate on reducing store miss and the improvement of LMC to reduce load miss.

Acknowledgements
This work is funded by National Science and Technology Development Agency (NSTDA) during 1997.

References

[1] J. Heinrich, " MIPS R4000 USER'S MANUAL", Prentice Hall, 1993.
[2] M. J. Charmey, T. R. Puzak, " Prefetching and memory system behavior of the SPEC95 benchmark suite", IBM Journal of Research and Development Volume 41, Number 3, May 1997, pp. 265-285.
[3] D. Alpert, D. Avnon, " Architecture of the Pentium Microprocessor", IEEE Micro, Vol 13 (3), June 1993, pp. 11-21.
[4] M. R.Zargham, " Computer Architecture single and parallel systems", Prentice-Hall, pp.145-182.
[5] K. I.Farkas, N. P.Jouppi, P. Chow, " How Useful Are Non-blocking Loads, Stream Buffers and Speculative Execution in Multiple Issue Processor?", Proceedings of the First IEEE Symposium on High-Performance Computer Architecture, 1995, pp. 78-89.