Available online at www.sciencedirect.com

SciVerse ScienceDirect

Physics Procedia 25 (2012) 1322 – 1328

Physics Procedia

2012 International Conference on Solid State Devices and Materials Science

Cache Performance Research for Embedded Processors

Chenxu WANG,Jiamin ZHENG,Mingyan YU

Microelectronics Center,Harbin Institute of Technology at Weihai,Weihai, Shandong

Abstract

The embedded processor performance is significantly influenced by cache whose performance depend on its architecture parameters. Meanwhile, In order to overcome the non-timing accurate flaw of software simulation method, In this paper, a hardware emulation method --RTL level models is used for CPU and cache controller, while circuit model for cache memory cell--is adopted to do research on cache performance . A more accurate design space, miss rate and cycle trend influenced by cache parameters, is presented. Compared with Round Robin, it shows that miss rate and cycle number of instruction cache is reduced by 22.08% and 20.36%, respectively, because Pseudo-LRU is adopted.

Keywords-cache; performance research; hardware model; replacement algorithm

1. Introduction

Embedded processor performance depends heavily on the performance of cache which depends on its architecture parameters. However, there are several parameters including the cache size, line size, associativity which will lead to a variety of cache configurations, and the cache performance varies according to the configuration of cache. So it's difficult for designer to implement a best configuration. Meanwhile, in relevant aspects of cache research, it shows that a software simulation method is still a mainstream of cache performance analysis. Nevertheless, there are many flaws of a software simulation method:

1.1 The pipeline structure of cache is not accurate enough, so it will be no delay for the next if the request of cache failed.

1.2 In the software simulator the miss request is sent to the lower level of memory hierarchy simultaneously when the dirty row has been replaced. However, in actual fact the two must be in different cycles.

1.3 In the software simulator the refill request has used additional cache port which may result in the same time, there are two fetch requests and one refill request.

 doi:10.1016/j.phpro.2012.03.239

1.4 Above all, the storage system of simulator is so simplified. Besides, the delay of memory access is a constant, there are great gaps between the real system and the simulator [1].

In the reference [2], [3] we see which relating to the cache design and exploration has arisen, but almost are based on simplescalar 2.0 simulators, Daniel Gracia Perez and Gilles Mouchard have drew a comparison among the simulators used for general application, and raise that the most relevant data to the present literature of architecture research has no reference value, because there're many defects of the simulator which can't be ignored.

Two replacement algorithm currently used is LRU and Round Robin, the implementation of LRU is complicated, high efficiency, and needs a lot of hardware support(in this paper the PLRU—Pseudo-LRU is used to replace the LRU), while the Round Robin is simple to implement and does not need complex hardware support by Sacrifice a little efficiency.

To solve these problems which have mentioned before, in this paper, a hardware emulation method -- RTL level models --is adopted to do research on cache performance, and we have compared the two replacement algorithm PLRU and Round Robin. This paper provides more accurate and reliable quantitative basis for the cache design of embedded processor. Furthermore, the conclusion of the paper will have a good reference for the cache design of processor which is executed in-order with single instruction issue.

2. Simulation

2.1 Cache hardware

In this paper, it supports a Harvard structural one level cache (two separate instruction cache and data cache), which allows independent selection and optimization of I-cache size, D-cache size, line size and associativity. The tag field and the data RAM is generated by memorycompiler, and the Following figure illustrates the structure of D-cache

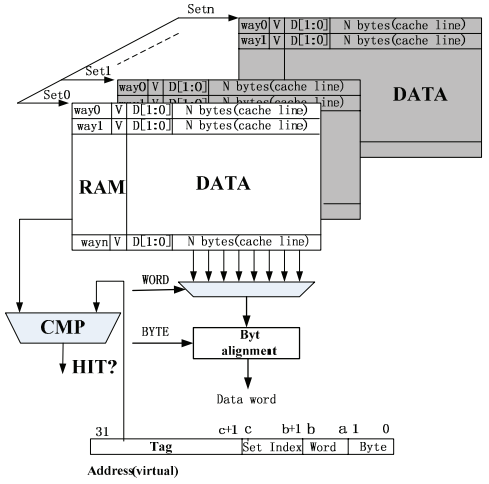

Figure 1. Data Cache Structure.

In this paper the cache has 32 address bits, the bits[31-(c+1)] are the tag field , the tag from the address is compared against the upper portion of the data RAM by comparator(CMP) to determine whether the entry in the cache corresponds to the request address; the bits[c-(b+1)] are used to index the cache; while the bits[b-a] are used to select the word for the multiword cache block and the least 2 bits of the address are

the byte offset bits. The figure above shows that Each way of a set contains N bytes (one cache line) and one valid bit. There also have two dirty bits for each line, one for the lower N/2 bytes and the other for the upper N/2 bytes.

2.2 Simulation platform

The simulation in this paper is based on a timing-accurate RTL level model, in order to improve the simulation accuracy, the method of combining the software with hardware is adopted to do research on cache performance. Figure 2 shows the structure of the simulation platform.

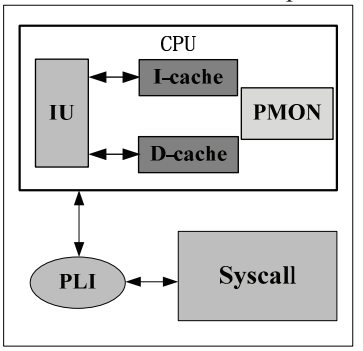

Figure 2. simulation platform structure.

The platform consists of three parts: the CPU core module, the PLI interface and the Syscall module. It shows that the CPU core is connected to the Syscall module via the PLI interface, and the system function in application program such as printf will be sent to the Linux system to be dealt with by the Syscall. From the Fig. 2, it shows a performance monitor (PMON) in the CPU core module, PMON collects and reads out some specific hardware statistical information that resembles performance factors such as I-cache, D-cache hit rate, I-cache, D-cache miss rate, cycle per instruction and so on, by which compiler writers, system developers and application programmers can analyze the performance limitations of their algorithms or codes on the CPU cores, and do optimization based upon the analysis result.

2.3 Simulation strategy

In this paper the study of cache design and exploration is based on the equation (1)

$$S_{cache} = N_{set} \times W_{line} \times N_{way} \quad (1)$$

S_{cache},N_{set},W_{line},N_{way} represent cache size, number of sets, line size and associativity respectively. The study of cache performance in this paper is divided into four cases:

1. When study the impact of cache size on cache performance, keep the parameters W_{line},N_{way} unchanged, only change the parameter N_{set}, so S_{cache} vary with N_{set}.
2. When study the impact of associativity on cache performance, remain the parameter S_{cache} and W_{line} unchanged, so the parameter N_{set} will change in the opposite direction according to the parameter N_{way}.
3. When study the impact of line size on cache performance, remain the parameter S_{cache} and N_{way} unchanged, so the parameter N_{set} will change in the opposite direction according to the parameter W_{line}.
4. When study the impact of PLRU and Round Robin on cache performance, keep the parameter S_{cache} and W_{line} as fixed value, and change the parameter N_{set} and N_{way}, so as to study the impact of these two replace algorithm on cache performance with different associativity.

3. Cache performance analysis

3.1 CPU core

The CPU core used in this paper is a super high performance and low power microprocessor core which is based on the pipelining design methodology. The main instruction pipeline of the core is partitioned to seven stages, elaborately designed and independent missions are performed at each pipeline stage to supply very high throughput for most RISC characteristic instructions and high main frequency for the processor. The basic processor includes: 4 set 32-way full associate BTB; Virtual address space: 4G-Bytes; Full associative 64-entry instruction TLB (ITLB) and 64-entry data TLB (DTLB); a Harvard structural one level cache (two separate instruction cache and data cache) which has two cycles access latency, the first cycle is used for tag comparison, and the second is used for data transmission between cache and CPU.

3.2 benchmark

Due to the simulation of the RTL level microprocessor, we need to run a performance appraisal system code to complete the evaluation. The SPEC2000 benchmark suite is the current defacto standard for simulation-based computer architecture research. In this paper the SPEC 2000 benchmark is used to evaluate the performance of cache. The largest input set for each benchmark in this suite is called the reference input set. Although this input set typically yields the most realistic behavior, it is rarely simulated to completion due to its very long simulation time. In this paper a fast-forward technique is adopted which is named as truncated execution [4]. The implementation is to perform detailed simulation for Y million instructions after fast-forwarding through the first X million instructions while tracking the simulation statistics for only the last Y million. The fast-forward X million instructions is the solution to "warm-up" the processor and memory before starting detailed simulation. Two variations of the technique are as follows X=20M, Y=100M. Meanwhile, we select 9 components of SPEC2000 as the test sets due to the impact of different component suite on the cache performance. (5 integer : gzip, vpr_place, vpr_route, gcc, mcf, 4 floating point : mesa, art, equake, ammp).

3.3 Experimental results analysis

1.1 The impact of associativity on cache performance

In this experiment the cache size is 16K and the line size is 32Byte, the associativity is from 2-way up to 64-way as 2-fold increase in trend. The experiment data is the average of the 9 benchmarks of SPEC2000 which is mentioned above. The results are illustrated in figure 3.

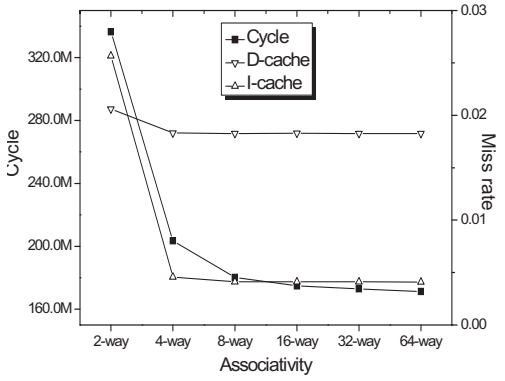

Figure 3. impact of associativity on miss rate and cycle.

The miss rate of I-cache, d-cache and the cycle decreased with the increasing of associativity. The cycle and the miss rate of cache have wide variations when the associativity is less than 4-way. However, the cycle and the miss rate of cache is in comparatively steady state when the associativity goes up from 4-way.

1.2 The impact of cache size on cache performance

The line size of cache is 32Byte and the associativity maintains a fixed value 32-way in the second experiment. The cache is at twice the trend growth from 2K to 64K. Experiment results can be seen below in figure. 4, 5 and 6.

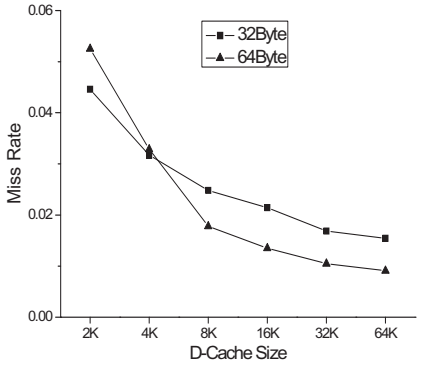

Figure 4. impact of D-cache size line size on miss rate.

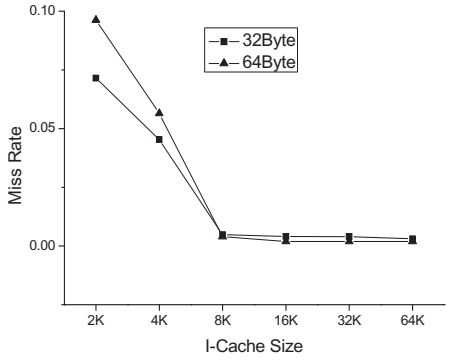

Figure 5. impact of I-cache size line size on miss rate.

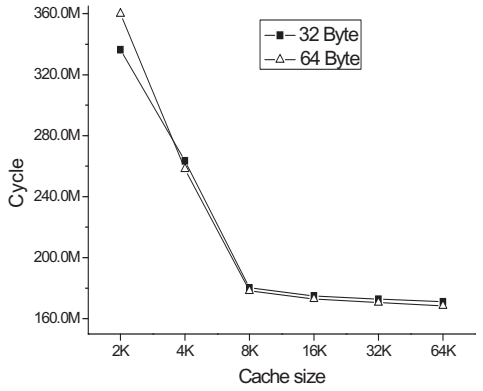

Figure 6. impact of cache size line size on cycle.

When line size is a fixed value, it is clear to see that the miss rate dropped due to the increasing of cache size. As cache size is small (less than 8K) the miss rate and the cycle have drastic variations. On the other hand, when cache size is more than 8K, the miss rate and the cycle change slightly and the miss rate has a similar trend to that of cycle.

1.3 The impact of line size on cache performance

The associativity of cache is 32-way and the line size is either 32Byte or 64Byte in this experiment. The Fig. 4, 5, 6 show the experiment results. According to the figure the miss rate and the cycle of 32Byte cache is below that of 64Byte when cache size is small, but when cache size grows it turn out to be the opposite. The 32Byte cache and 64Byte cache intersect at 4K for D-cache which is illustrated in Fig. 4, besides, the two intersect at 8K for I-cache illustrated in Fig. 5.

1.4 The impact of replacement algorithm on cache performance

In the last experiment the parameter is set as follows, Cache size is 16K, line size is 32Byte, and the associativity goes up from 2-way to 64-way as 2-fold increase in trend. Figure7 and figure8 show the statistics on the experiment data.

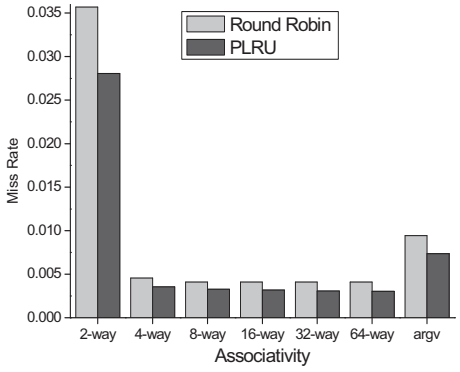

Figure 7. impact of PLRU and Round Robin on miss rate.

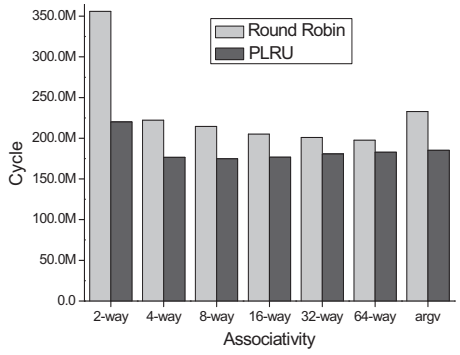

Figure 8. impact of PLRU and Round Robin on cycle.

They show that the miss rate of cache and the cycle per instruction is decreased by 22.08% and 20.36% respectively for PLRU, compared with Round Robin. Besides, there is a fine difference about the impact of PLRU on the cycle and the miss rate, the cycle number of instruction has a slight increase as the associativity get higher, the reason is that the historical information of cache access will be stored by PLRU algorithm, which consumes large amounts of hardware resources, and therefore, it influence the access time of cache and increase the cycles per instruction.

4. Conclusion

In this paper a hardware emulation method is adopted to analyze the cache performance, the experiment data is accurate and reliable due to the high precision RTL-level module and adequate simulation platform and benchmarks. A more accurate design space, miss rate and cycle trend influenced by cache parameters, is presented via experiment, which will have grate use for reference of cache design on the space limitations for low power embedded systems and the implementation of single-instruction issue in-order execution microprocessor. Moreover, by analyzing and comparing a great deal of experiment data of two replacement algorithm PLRU and Round Robin, we get to the conclusion that Compared with Round Robin, the miss rate and the cycle per instruction of cache is reduced by 22.08% and 20.36%, respectively, when Pseudo-LRU is adopted.

References

[1] Daniel Gracia Perez, Gilles Mouchard, Olivier Temam. MicroLib, “A Case for the Quantitative comparison of micro-architecture mechanisms” Proceedings of the 37th international symposium on microarchitecture. PoRTLand, Oregon, USA, 2004.12, pp.43- 45.

[2] Chaitali Chakrabarti. “cache Design and Exploration for Low Power Embedded systems” ieee international conference on performace, computing, and communications. phoenix, AZ, USA, 2001.4, pp.135-139.

[3] Wen-Tsong Shiue, Chaitali Chakrabarti. “Memory Design and Exploration for Low Power Embedded Systems”. Journal of VLSZ signal processing, 1999, 22(1), pp.281-290.

[4] Joshua J. Yi, Sreekumar V. Kodakara, Resit Sendag, David J. Lilja, Douglas M. Hawkins “Characterizing and Comparing Prevailing Simulation Techniques”. High-Performance Computer Architecture, 2005, pp.266-277

[5] Joshua Jeffrey Yi. “Improving Processor Performance and Simulation Methodology”. Minnesota: The faculty of the graduate school of the university of Minnesota, 2003.12.