Anticipatory Access Pipeline Design for Phased Cache

[1]Chih-Wen Hsueh, [1]Jen-Feng Chung,
[1]Department of Electrical and Control Engineering
National Chiao Tung University
Hsinchu, Taiwan
jfchung.ece88g@nctu.edu.tw

[2]Lan-Da Van, and [1,2]Chin-Teng Lin, *IEEE Fellew*
[2]Department of Computer Science
National Chiao Tung University
Hsinchu, Taiwan
ldvan@cs.nctu.edu.tw

***Abstract*—**For an embedded processor, the cache design almost occupies half chip area and power consumption. According to Amdahl's law, if the power consumption of cache memories is reduced, the embedded processor can significantly save much power. However, the cache misses result in the penalty of thousands of cycles waiting and power consumption due to increasing the number of external memory access. Based on the above reason, the phased cache design is proposed and can largely improve the power consumption which wastes a set-associative cache. In this paper, the embedded pipelining processor without stalling and low-power phase cache is practiced with high-level simulation to achieve high-performance and low-power design. As experimental results, the proposed phase cache can reduce 44% power consumption compared with traditional one-access-cycle cache and eliminate pipeline stalls incurred by phased cache with only 6% gate count overhead.

I. INTRODUCTION

Caches take nearly half area and power consumption of a microprocessor. Like the microprocessor ARM920T [1], caches consume 44% power of this embedded processor. According to Amdahl's law, if the power consumption of cache can be reduced, the processor can significantly save much power.

The cache is one of the most attractive targets for power reduction. There are several techniques [2]-[4] for reducing the power consumption of on-chip set-associative caches. The structure of a conventional cache and phased cache is shown in Fig. 1. Due to the increasing of cache sets, it also increases the power consumption of cache access. For the architecture of a phased *n*-way set-associative cache, the tag value of each set is accessed to compare with the tag bits of memory address in the first cycle. If the data value of hit set detected by phased 1 is fetched in the second cycle, other sets will be disabled to reduce power consumption. The architecture of sentry-tag and halting-tag caches can filter out the unnecessary set activities from comparison with lower tag bits before cache access [5-6]. This method of the way-predicting set-associative cache can predict the hit information by most resent used algorithm [7] and reduce power consumption by accessing the predicted set instead of accessing all of the sets.

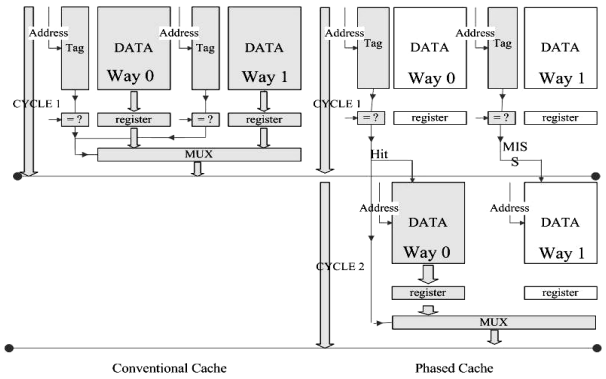

Figure 1. Structure of conventional cache and phased cache.

Although the phased cache can reduce most power consumption, it requires two access cycles which is more than conventional one-access-cycle cache. The architecture of combining with sentry-tag and way-predicting set-associative cache can reduce power consumption in one cache access cycle. However, these methods also pay the trades-offs for power consumption, cost and complication. On the other hand, a phased cache design may cause a data hazard if there is a data dependency. It will generate a pipeline stall and reduce the performance. Phased cache design only impacts the performance of data cache. Therefore, this paper proposes a pipeline processor design to speed up the data cache access and to solve the pipeline stall due to phased cache design.

II. ANTICIPATORY ACCESS PIPELINE DESIGN FOR PHASED CACHE

A new method for phased cache improvement is proposed in this paper. The anticipatory access pipeline design and phased cache are combined to eliminate the cache delay. We take the tag comparison phase of data cache ahead by one stage. The anticipatory pipeline design with phased cache can reduce the unnecessary cache power consumption and obtain the hit data at the same time.

978-1-4244-1684-4/08/$25.00 ©2008 IEEE

The anticipatory access pipeline for phased cache and modified phased cache is shown in Fig. 2 (a) and (b), respectively. In order to gain the cache address early, a cache addressing calculator is added in the register file stage. In this way, the tag of Level 1 (L1) cache can be accessed and compared with memory address in the arithmetic and logic stage, and then the instruction performs the L1 data cache access in the write-back stage. For example, given three memory addressing modes of general processors [8] are listed in Table I. Their utility rates for these three addressing modes are from 70% to 90% in performing *Gcc* and *Tex* programs.

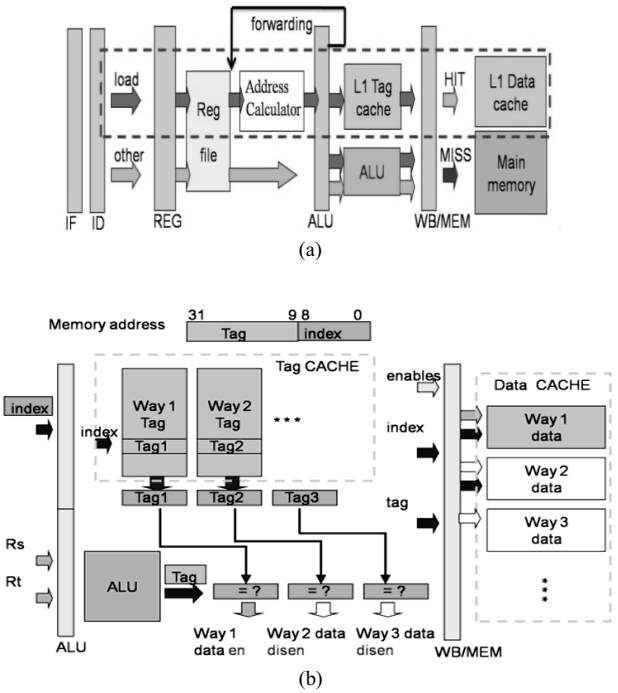

Figure 2. (a) Anticipatory access pipeline design for phased cache and (b) Phased cache architecture in Anticipatory access pipeline.

TABLE I. MEMORY ADDRESSING MODE

Addressing Mode	Example	Operation	Utility Rate	
			Gcc	Tex
Immediate	LD R4,#3	R4<-Mem[3]	36%	43%
Placement	LD R4,3(R1)	R4<-Mem[R1+3]	40%	32%
Register Indirect	LD R4,(R1)	R4<-Mem[R1]	11%	24%

The algorithm of anticipatory access pipeline design with phased cache is shown in Fig. 3. The architecture of pipeline design has five stages including the instruction fetch (IF), the instruction decode (ID), the register file (REG), the arithmetic and logic unit (ALU), and the write-back and memory (WB/MEM). In the ID stage, an instruction is decoded and detected whether the instruction operation is memory access or not. In the REG stage, the register values are read. Besides, if the operation is memory access, the memory address is calculated by each addressing mode in this stage. In the ALU stage, a tag cache can be accessed if there is an instruction of memory load. After the access of the tag cache, the tag value of each set will be compared with the full memory address calculated by the ALU unit, and then the data cache of hit set decided by tag comparison is enabled in the WB/MEM stage.

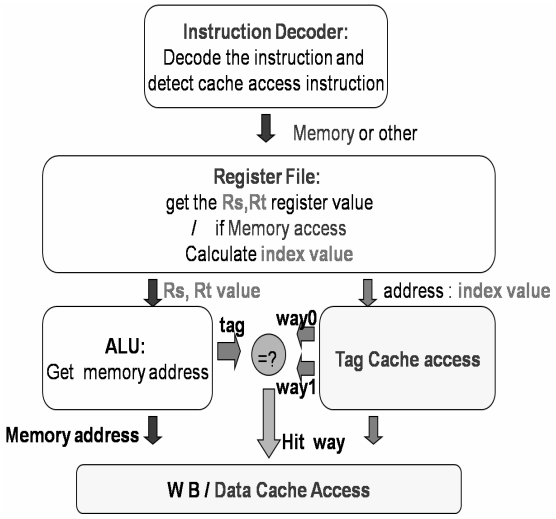

Figure 3. The algorithm of phased cache with pipeline improvement.

The phased cache access still requires two cycles, but the proposed phased cache with anticipatory access pipeline design can perform the tag comparison phase one cycle earlier and fetch the hit data in the same stage compared with conventional cache. The proposed phased cache can eliminate the pipeline stall caused by the phased cache. Besides, if L1 cache is missed, the phased cache can be detected by tag comparison in the ALU stage. With the miss information, the pipeline can access the next level memory like L2 cache or main memory one cycle early. For example, L1 cache requires 1 cycle and the next level memory required 4 cycles to access data. If L1 cache is missed, it only requires 4 cycles to access data in the proposed design. Hence, the proposed design, which is compared with the conventional architecture required 5 access cycles, can fetch the required data one cycle earlier.

There are many difficulties caused by the modification of pipeline architecture. For example, critical paths and pipeline hazards will be the risk of this design. To shorten the critical path, the calculation of cache address (index value only) in the REG stage is adopted. The index value for cache access is calculated by the REG stage, and the tag value is compared by the ALU stage.

A forwarding path from the REG stage to the REG stage to solve the data hazard is demonstrated in Fig. 4. The *load* instruction in the REG stage may cause a data hazard of the r1 register on Cycle n+1. The lower bits of r1 will be calculated in the REG stage for the forwarding operation. The pipeline design with phased cache in a RISC processor is implemented to verify the feasibility and to test the other hazards of proposed architecture.

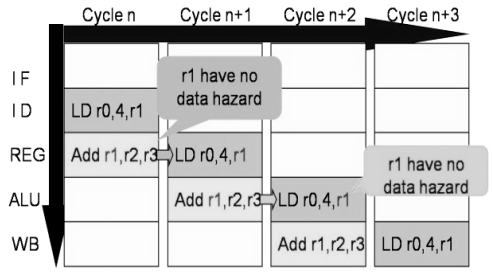

Figure 4. Forwarding path for anticipatory access pipeline.

TABLE III. SEVEN REANDOM BENCHMARKS FROM SPEC95 AND SPEC2000

Benchmarks	SPEC	Description
compress95	CINT95	A in-memory version of the common UNIX utility.
li95	CINT95	Xlisp interpreter.
gzip	CINT2000	Compression
gcc	CINT2000	C Programming Language Compiler
ammp	CFT2000	Computational Chemistry
parser	CINT2000	Word Processing
bzip2	CINT2000	Compression

III. ANALYSIS OF POWER AND EXECUTING TIME

The set activities in each cache access are used to express the ratio of power consumption. As shown in (1), the activities of conventional cache (ACT*convention*) and the activities of proposed design (ACT*design*) are the average set activities in each access. The activity ratio is expressed by the percentage of power consumption between conventional cache and proposed design.

$$Activity\ Ratio = (\mathrm{ACT}design) / (\mathrm{ACT}convention). \quad (1)$$

$$\begin{aligned} Power\_cache = {} & Hit\_ratio \times Way \times (Tag\_power \times H\_T\_enable\_ratio + \\ & Data\_power \times H\_D\_enable\_ratio) \\ & + Miss\_ratio \times Way \times (Tag\_power \times M\_T\_enable\_ratio + \\ & Data\_power \times M\_D\_enable\_ratio) + Other, \end{aligned} \quad (2)$$

where *H_T_enable_ratio* is the activity ratio of tag cache in hit situation, *H_D_enable_ratio* means the activity ratio of data cache in hit situation, *M_T_enable_ratio* is the activity ratio of tag cache in miss situation, and *M_D_enable_ratio* means the activity ratio of data cache in miss situation.

There are two possible results in each cache access as *hit* or *miss*. Besides, the area parameters (tag, data) of tag and data cache are included in detailed analysis. The power consumption in each access can be expressed in terms of the number of cache ways (Way), the hit ratio, and the miss ratio as shown in Table II. For example, we calculate the result of power analysis for a 4-way, 8-bit tag, and 32-bit data cache. As the result, the phased cache can reduce 61% power consumption compared with conventional cache.

TABLE II. POWER ANALYSIS OF POWER CONSUMPTION BETWEEN CONVENTIONAL CACHE AND PHASED CACHE

Power consumption	Conventional cache	Phased cache
Power of hit	Way(tag + data)	Way(tag)+data
Power of miss	Way(tag + data)	Way(tag)
Total power consumption	Way(tag + data)	HR{Way(tag)+data}+ MR{Way(tag)}
Power (4-way, 95% HR)	100%	39%

The simulator *SimpleScalar* [9] is used in all experiments to simulate a number of benchmarks for a 4-way and 4KB cache. Seven benchmarks shown in Table III from SPEC95 and SPEC2000 are randomly selected. By the simulator, the power reductions with the design of 2-way, 4-way, and 8-way phased caches are 42%, 61%, and 71%, respectively. This result shows the proposed phased cache scheme can reduce the access of cache efficiently.

We estimate the performance for phased cache with and without anticipatory access pipeline design. Fig. 5 shows the improvement of performance with different benchmarks. As the result of the proposed pipeline phased cache design, the average executing cycles of benchmarks can be reduced to 8%. As the results shown in Fig. 6, on average 38% cache access cycles caused by phased cache would be eliminated because the phased cache would not generate data hazards for each cache access. If there is any data dependency, phased cache would stall the pipeline operation. Hence, there are about 38% load instructions with data dependency, and 38% performance of cache access can be improved in average.

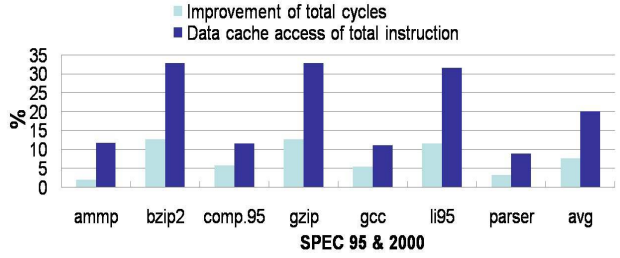

Figure 5. Improvement of phased cache with pipeline design in different benchmarks.

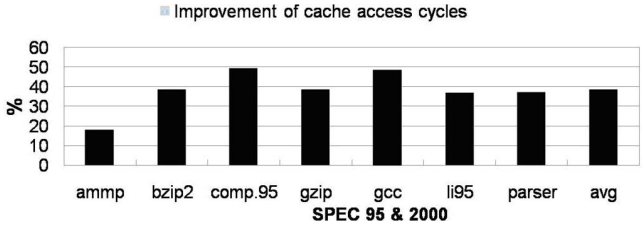

Figure 6. Improvement of cache access cycles.

The performances of cache designs are shown in Fig. 7. The anticipatory access pipeline with phased cache architecture can reduce 60% power consumption of a 4-way cache and 38% cache access cycles. Fig. 8 shows the pipeline architecture of phased cache with anticipatory access pipeline design and conventional cache with anticipatory access pipeline design. The performance and power consumption of them is listed in Table IV. The anticipatory access pipeline design proposed in this paper can reduce 4% to 5% the total of executing cycles. Phased cache with proposed pipeline design can reduce 60% power consumption without any timing penalty. Therefore, high performance access pipeline design combined with phased cache can induce the greatest benefit.

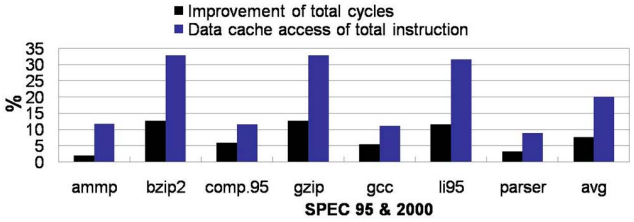

Figure 7. Improvement of phased cache with pipeline design in different benchmarks.

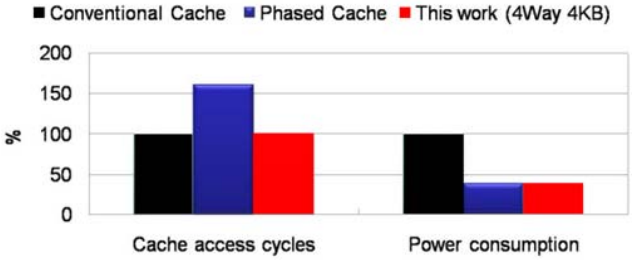

Figure 8. Performance of conventional cache, phased cache and this work.

TABLE IV. THE PERFORMANCE AND POWER CONSUMPTION OF EACH ARCHITECTURE

4-way cache	Phased cache	Anticipatory pipeline	Power (%)	Cycles (%)
Conventional cache	×	×	100%	100%
Phased cache	O	×	40%	108%
Anticipatory Pipeline	×	O	100%	94%
This work	O	O	40%	99%

The high power reduction of phased cache about 44% compared with other traditional one-access-cycle caches is maintained with only 6% gate count overhead. The increment of 6% gate count is from anticipatory access pipeline design in the REG stage. Although the area is increased, it will not cause the decline of performance.

IV. CONCLUSIONS

The phased cache with anticipatory access pipeline design has been proposed in this paper and is able to save, across different benchmark suites, an average 60% of the energy of a conventional 4-way set associative cache without any timing penalty. The important key feature is the combination of anticipatory access pipeline design and phased cache. Thus, the pipeline stalls caused by phased cache is eliminated, and the high power reduction of phased cache about 44% compared with other traditional one-access-cycle caches is maintained with only 6% gate count overhead.

ACKNOWLEDGMENT

This work was supported in part by the National Science Council (NSC), Taiwan, R.O.C., under Grant NSC-96-2221-E-009-220, NSC-96-2220-E-009-038, and NSC-96-2221-E-009-058.

REFERENCES

[1] Simon Segars, "Low Power Design Techniques for Microprocessors," *ISSCC 2001*, Feb 4th 2001.

[2] Ching-Long Su, Alvin M. Despain, "Cache design trade-offs for power and performance optimization: a case study," *International Symposium on Low Power Electronics and Design*, pp. 63-68, 1995.

[3] J. Kin, Munish Gupta, and Mangione-Smith, "The filter cache: an energy efficient memory structure," *Thirtieth Annual IEEE/ACM International Symposium on Microarchitecture,* vol. 1, pp.184-193, Dec. 1997.

[4] Hoon Choi, Myung-Kyoon Yim, Jae-Young Lee, Byeong-Whee Yun, and Yun-Tae Lee, "Low-power 4-way associative cache for embedded SOC design," *The 13th Annual IEEE International ASIC/SOC Conference*, 13-16, pp.231-235, Sept. 2000.

[5] Yen-Jen Chang, Shanq-Jang Ruan, and Feipei Lai, "Design and analysis of low-power cache using two-level filter scheme," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 11, pp.568-580, Aug. 2003.

[6] Chuanjun Zhang, F. Vahid, Jun Yang, and W. Walid, "A Way-Halting Cache for Low-Energy High-Performance Systems," *Computer Architecture Letters*, vol. 2, pp. 5-5, Jan. 2003.

[7] K. Inoue, T. Ishihara, and K. Murakami, "Way-predicting set-associative cache for high performance and low energy consumption," *International Symposium on Low Power Electronics and Design*, pp.273-275, 1999.