Low Power Set-Associative Cache with Single-Cycle Partial Tag Comparison

Jian Chen, Ruihua Peng, Yuzhuo Fu
School of Micro-electronics, Shanghai Jiao Tong University, Shanghai 200030, China
{chenjian, pengruihua, fuyuzhuo}@ic.sjtu.edu.cn

Abstract:

This paper presents a novel structure for partial tag comparison cache. By triggering partial comparison unit and data sub-banks with inversed clock, the partial comparison can be activated before cache access, leading to single-cycle hit time. Meanwhile, it consumes less energy in each cache access than that of conventional cache by filtering out unnecessary tag and data array accesses. Simulation results show the proposed cache achieves an average reduction on power consumption of 24.8% compared with predictive cache.

I. Introduction

Set-associative caches are widely used in modern microprocessor designs to enhance the performance by exploring temporal and spatial localities. Compared with other cache organizations, set-associative cache provides a better trade-off between miss-ratios and complexities of implementations [1]. However, traditional set-associative caches have their drawbacks in terms of power consumption. As illustrated in Figure 1, a typical 4-way set-associative cache access has to go through a series of operations: selecting cache line, reading tag and data array, performing tag comparisons, and driving the multiplexer select signals. These operations are inefficient from the perspective of power consumption, because 3 out of 4 data reads and tag reads are useless.

Various techniques have been proposed to address the problem. Phased caches [2] separate the cache access into two phases. Tag array is accessed and compared in the first phase. In the second phase only the hit data way is accessed, resulting in less data way access energy at the cost of longer cache access latency. Way-predicting cache attempts to predict a way where the required data may be located before accessing tags [3]. If the prediction is correct, no more tag array access is required and the access latency is similar to that of direct-mapped cache of the same size. On the other hand, if the prediction fails, tag comparison has to be performed leading to extra access time and power dissipation than that of conventional set-associative cache. Partial address comparison was brought into cache design world by Lishing Liu [1]. It was initially intended to produce fast and accurate predictions of cache coordinates for array access. Rui Min, etc [4] explored the power saving aspect of partial comparison by selectively disabling the sense amplifiers in data array. However, directly disabling sense amplifiers in data array is not generally available in ASIC designs, which severely limits its applications.

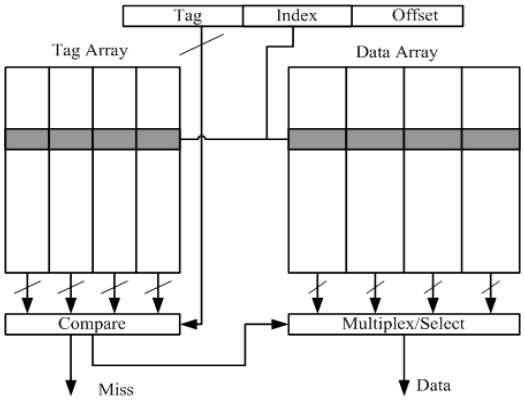


Figure 1. The diagram of a 4-way set-associative cache

This paper proposes a novel partial tag comparison cache structure with inverse-phased clock. It has a single cycle hit time, and achieves more power savings compared with traditional way-predictive cache designs. It is also more applicable in ASIC designs than that in [4], using generally available enable signals of memory banks. The rest of the paper is organized as follows: Section II describes the detailed architecture of the proposed partial tag comparison cache. Section III presents power and performance analysis for the proposed cache. Section IV provides simulation results. Section V summarizes the paper.

II. Single-Cycle Partial Tag Comparison Cache Structure

Rather than compare the whole tags from the tag array in conventional set-associative caches, partial tag comparison cache compares the Least Significant Bits (LSB) of the tag first. The idea behind this can be traced to the observation that the difference of tags in the same cache line is centered on LSBs of tags due to the program locality. By comparing these bits, useless tag and data array accesses can be effectively reduced, leading to significant power reduction. Section II.A presents the detailed architecture to implement this idea. In section II.B, the inversed clock technique is introduced to prevent hit time degradation.

A. Architecture of Partial Tag Comparison Cache

Figure 2 shows the architecture of a 4-way partial tag

comparison cache. Compared with the conventional cache, there is an additional register file in the proposed cache structure. Each row of the register file contains four blocks, which are associated with corresponding tags respectively. Each block has three bits, one for valid bit, the other two for partial tags. Hence, the Partial Comparison Width (PCW) of this cache is 2 bits. In addition, tag array and data array are divided into four sub-banks respectively. Each of the sub-banks can be enabled or disabled by CE (Chip Enable) signals, which is generally available in memory banks. The CE signals of the tag sub-banks come from the partial comparison unit, while those of the data sub-banks come from the tag comparison.

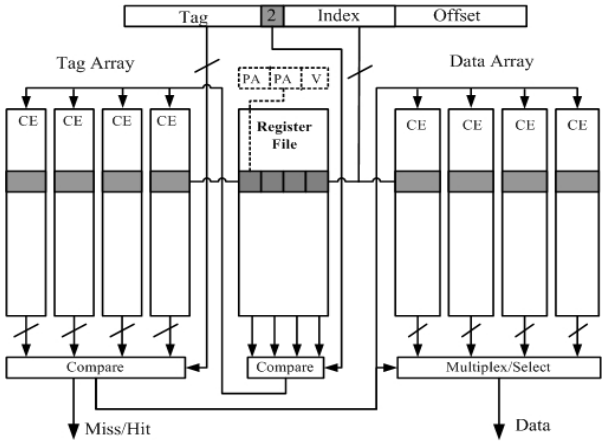


Figure 2. The structure of the proposed partial tag comparison cache

Each access of the proposed cache involves three steps. During the first step, the least 2 significant bits of the tag in the address are compared with the data read out from the register file. In the second step, two possible operations could be performed according to the results of the previous one. If partial comparison indicates no match or the matched block is invalid, then cache miss occurs. Under this circumstance, no tag and data array access happens. We call it Early Miss Detection (EMD). On the other hand, if there is one or more matches, the rest of the tag bits have to be compared to determine which way is a real hit. However, only the tags that are partially matched are accessed with the help of CE signal generated in previous step. Unnecessary tag accesses are filtered out to save energy. If there is a real hit, the access to the hit data sub-bank is activated in the third step. Otherwise, cache miss occurs with no extra data array accesses.

B. Timing with the Inversed Clock

Although the access of the proposed cache is divided into three steps, the first step can be effectively hidden in the clock cycle before tag and data access begins. The second and third steps can also be accomplished within one clock cycle. These operations are achieved by making the register file and the data array triggered by an inverse clock, which has 180° phase difference with the original clock.

The timing of the operations with this inversed clock is illustrated in Figure 3, where CLK represents the original clock and $\overline{\text{CLK}}$ is the inversed clock. Before the index field of cache address is used to access the tag and data arrays, it is latched into the register file by the positive edge of inversed clock. Partial tag access and comparison are completed before the positive edge of CLK. As a result, the bank enable signal can be used to enable the corresponding sub-banks in tag array in the following clock cycle. The similar operation applies for the data array. The tag access and comparison are completed within half clock cycle, leaving another half cycle for the data access. As a result, with inversed clock it can achieve single-cycle cache access from the perspective of processor.

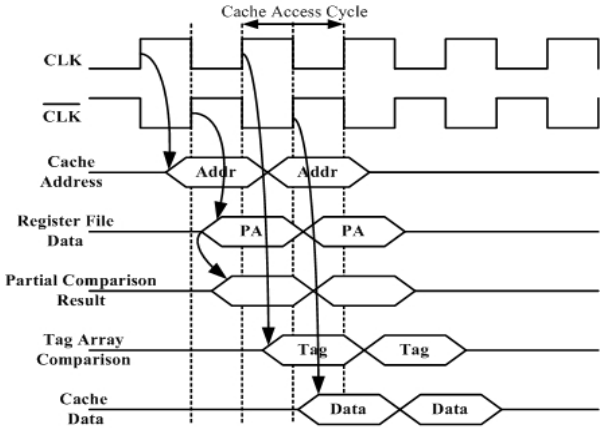


Figure 3. The timing diagram of partial tag comparison with inversed clock

III. Performance and Power Consumption Analysis

In terms of hit time, the performance of the proposed cache is superior than that of predictive cache, because the former has single cycle hit time while the later requires an average hit time of more than 1 clock cycle [3][5]. Besides, the partial comparison cache also has a lower average miss penalty due to the EMD feature. When EMD happens, the miss penalty is reduced by 1 clock cycle compared with that of the conventional cache. All of these factors contribute to a smaller average access time than the predictive cache with the same configurations.

However, the inversed clock technique imposes several constraints on cache design. The cache address has to be available before the negative edge of the original clock, leaving only half clock cycle for address preparation. It also sets an upper limit on the clock frequency, since the tag array access, together with tag comparison, has to be accomplished within half clock cycle. The tag path delay, however, can be reduced by increasing the partial comparison bit width to relieve the tag access and comparison work load. Table I shows the tag path delay versus the tag bit number with TSMC 0.18um generic process. As can be seen, the total delay on tag path reduces as the tag bits decreases. With these delays, the maximum speed the cache can achieve is around 200 MHz, which is suitable for embedded applications.

Table I. Delay on Tag Path vs. Tag Bit Number

	24 bits	21 bits	18 bits	15 bits
Tag Access	1.353ns	1.330ns	1.308ns	1.301ns
Comparator	1.150ns	1.150ns	1.150ns	1.010ns
Total	2.503ns	2.480ns	2.458ns	2.311ns

Although the proposed inversed clock technique has some negative influences on operating frequency of the cache, it can significantly reduce cache power consumption in two folds. On one hand, partial comparison prevents power consumption of unnecessary tag array accesses and comparisons. On the other hand, only the hit data sub-bank is accessed, which further saves the energy. Consider an n-way set-associative partial comparison cache. Let Nt be the average tag sub-bank access number. The total average power consumption of the cache can be written with the following formula:

$$P_{cache} = P_{reg} + P_{tag} \bullet N_t + P_{data} \bullet \text{Hit Rate} \quad (1)$$

where P_{reg}, P_{tag}, P_{data} represent the average power consumption of the register file, the tag sub-bank and the data sub-bank respectively. For the cache with fixed cache size, block size and set-associativity, $P_{data} \bullet \text{Hit Rate}$ is the minimum possible data access power. On the other hand, Nt, P_{reg} and P_{tag} are closely related to PCW. Generally speaking, Nt decreases as PCW increases because more bits are provided to differentiate distinct tags. P_{tag} also gets smaller as PCW increases since the width of tag set is reduced. However, P_{reg} increases with a faster slew rate than that of P_{tag} because the reduced bits on each tag set are combined together in the register file. Trade-off has to be made between P_{reg} and $P_{tag} \bullet Nt$ by selecting a proper PCW, so as to achieve minimum cache power dissipation. There are also some other factors that influence Nt, such as cache replacement policy and the program itself. But they are proved to have no significant influence on Nt as that of PCW.

IV. Simulation Results

This section presents the simulation results of the proposed cache structure with respect to power dissipation and performance. The simulation is based on SPEC95 CINT benchmarks. We modified SimpleScalar to capture the cache access activities. Cacti3.0 is also employed to estimate the power consumption of data and tag sub-banks [6].

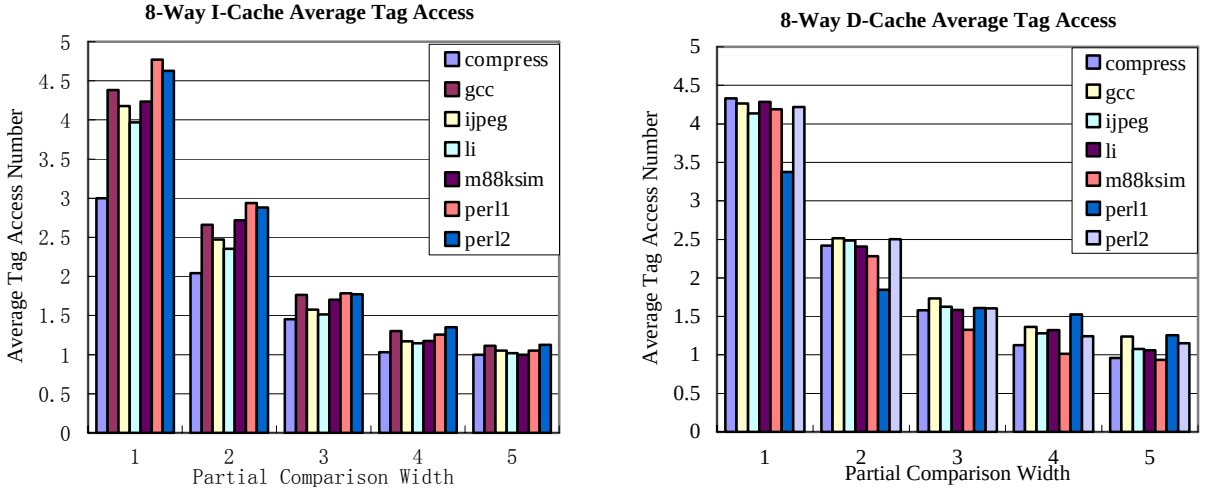
Table II. Average Time to Know Miss (cycle)*

	compress	gcc	ijpeg	li	m88ksim	perl
2-way	0.058	0.214	0.070	0.079	0.064	0.103
4-way	0.144	0.388	0.274	0.173	0.169	0.288
8-way	0.228	0.636	0.507	0.317	0.229	0.557

* 8KB I-Cache with 3-bit PCW.

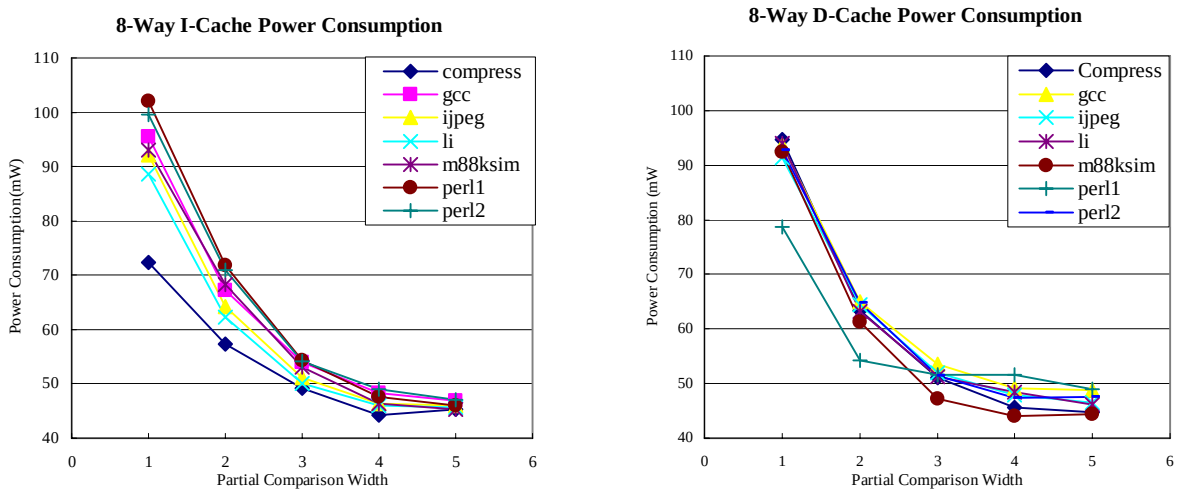
Table II illustrates the average time for partial comparison cache to know a miss, which directly contributes to cache miss penalty. All the data are smaller than 1 due to the zero latency of EMD. However, phased cache and predictive cache have 1 cycle and 2 cycles respectively to know a cache miss. Consequently, the proposed cache delivers a lower miss penalty than it counterparts.

Figure 4 shows the average tag access numbers with different PCW. As can be seen, when PCW approaches 5 bits, the average tag access number is almost one-way per access. It means most unnecessary tag accesses are eliminated in the 8-way set-associative cache. Figure 5 illustrates the corresponding power consumption. While the average power is reduced dramatically when PCW goes from 1 to 3 bits, no significant power reduction is observed when it is larger than 4 bits. As a result, 3 or 4 is the desired PCW according to power consumption. The average power consumption between different caches is also listed in Table III. Among the three types of caches, both partial comparison cache and predictive cache demonstrate less power consumption than



(a) (b)

Figure 4. Instruction cache and data cache average tag access number. Both caches have a size of 64KB, block size of 16 bytes. Both caches employ LRU replacement policy.



(a) (b)

Figure 5. Instruction cache and data cache average power consumption. Both caches have a size of 64KB, a block size of 16 bytes. Both caches employ LRU replacement policy.

Table III. Average Power Consumption of Different Caches*

	Partial Comparison Cache (mw)			Predictive Cache (mw)			Phased Cache (mw)		
	2-way	4-way	8-way	2-way	4-way	8-way	2-way	4-way	8-way
compress	12.435	16.187	22.992	11.291	16.774	27.866	17.712	36.764	76.216
gcc	12.278	16.482	24.938	11.894	18.649	32.401	17.712	36.764	76.216
ijpeg	12.689	16.482	23.982	14.276	26.066	50.341	17.712	36.764	76.216
li	12.423	16.199	23.855	12.292	19.889	35.402	17.712	36.764	76.216
m88ksim	12.674	17.081	24.024	11.778	18.290	31.534	17.712	36.764	76.216
perl	12.186	16.180	24.641	10.849	15.395	24.532	17.712	36.764	76.216

*The PCW is 3 bits. All the caches have a size of 64KB, block size of 16 bytes, and use LRU replacement.

phased cache with similar configuration. In addition, the proposed partial comparison cache has more consistent power consumption performance than predictive cache since prediction algorithm is sensitive to benchmarks. The advantage of partial comparison cache becomes more evident as the set-associativity gets higher. In 8-way configuration, it has an average reduction on power consumption of 24.8% compared to the corresponding predictive cache.

V. Conclusion

This paper proposes a single-cycle low power partial comparison cache. It uses partial tag comparison to achieve high power efficiency. It also employs inversed clock to compact the hit time into 1 clock cycle. Simulation results show that it not only has low power dissipation, but also has consistent performance in power consumption with different benchmarks. Consequently, it can be applied in embedded SoCs for specific applications as well as low power general purpose processors.

Acknowledgement

The research is partially supported by OnSemi Foundation and National 863 High-tech Program No. 2003AA1Z1350.

References:

[1]. Lishing Liu, “Partial Address Directory for Cache Access” IEEE Trans. On VLSI Systems, Vol.2, No.2, June 1994.

[2]. C. Su and A. Despain, “Cache design tradeoffs for power and performance optimization: A case study,” in *International Symposium on Low Power Electronics and Design*, pp. 63– 68, 1997.

[3]. Brad Calder, Dirk Grunwald, Joel Emer, “Predictive Sequential Associative Cache”，the 2nd IEEE International Symposium on High Performance Computer Architecture, San Jose, pp 244-254, Feb. 1996.

[4]. Rui Min, Zhiyong Xu, Yiming Hu, etc., “Partial Tag Comparison: A New Technology for Power-Efficient Set-Associative Cache Designs”, Proceedings of the 17th International Conference on VLSI Design, 2004

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

[6]. Premkishore Shivakumar, Norman P. Jouppi, “CACTI 3.0: An Integrated Cache Timing, Power, and Area Model ”, Western Research Laboratory, Research report 2001.2.

Key Words:
Partial Tag Comparison, Set-associative Cache, Inversed Clock