Comparing Multiported Cache Schemes

Smaïl Niar
University of Valenciennes, France
Smail.Niar@univ-valenciennes.fr

Lieven Eeckhout **Koen De Bosschere**
Ghent University, Belgium
{leeckhou,kdb}@elis.rug.ac.be

Abstract

The performance of the data memory hierarchy is extremely important in current and near future high performance superscalar microprocessors. To address the memory gap, computer designers implement caches to reduce the high memory latencies that are observed in the processor. Due to the ever increasing instruction window sizes and issue widths in new microprocessor designs, designers will need to implement multiported caches in order to be able service multiple memory requests per clock cycle. Several studies have been conducted in previous work dealing with the performance in terms of Instructions Per Cycle (IPC) for such cache designs. However, little is known about the access time, the energy consumption and the required chip area for multiported caches. This paper quantifies these aspects for several multiported cache designs.

Keywords: computer architecture, cache memories, multiported caches, multibanked caches

1. Introduction

It is well known that the processor/memory speed gap increases as VLSI chip technology continues to clock microprocessors at ever higher frequencies [10]. As a consequence, accessing memory might take several hundreds of processor cycles. As such, the memory access latency easily becomes the bottleneck of the system. Caching and prefetching are two well known solutions that can reduce (or hide) memory latencies.

In modern superscalar processors that use dynamic and highly speculative instruction scheduling, a large number of instructions can be issued per cycle. In order to sustain this high instruction throughput, the bandwidth to the data memory hierarchy should be scaled accordingly. For example, Sohi and Franklin [7] report that to attain an IPC of 16, 6 requests to the data cache must be supported per cycle. In other words, the data cache must be capable to service several requests simultaneously. In existing designs this is attained by implementing multiple cache ports (like in the MIPS R10000 [11]) or by using time division multiplexing (like in the DEC Alpha 21264 [3]). The latter technique accesses the cache multiple times per clock cycle. Consequently, as clock cycles keep on shrinking, it can be expected that this approach will not scale well. In case of multiported cache designs, we can make the following classification: true multiported caches, multibanked caches and hybrid multiported-multibanked cache designs.

In this paper, we are studying the organization of multiported caches. In contrast to previous work that mainly focused on the performance of multiported caches in terms of instructions per cycle (IPC) [2,5,9], we evaluate several multiported cache designs in terms of access time, power consumption and chip area. To the best of our knowledge, very little research was done previously on these aspects. In this paper, we evaluate the following schemes: a true multiported cache, two multibanked cache designs with block interleaving and word interleaving, respectively, and two hybrids schemes: the locality-based interleaved cache and the multibanked cache with multiple ports per cache bank.

These five schemes have been chosen because they represent the basic structures for a multitude of other schemes. And their performances can be used to approximate the performance of any other data cache scheme.

2. Multiported caches

This section presents the existing flavors of multiported caches: true multiported caches, multibanked caches and hybrid cache designs.

2.1. True multiported cache

In a true multiported cache (or perfect cache "PRF"), see Figure 1.a, there is only one structure containing all the data (plus tags). A true multiported cache has P access ports that operate independently from each other. This means that multiple accesses to the cache can operate to any cache block in the same clock cycle. True multiported caches offer the highest possible access rate per cycle but are complex, and thus slow and costly. Therefore, this solution can only be used for small memories (such as register files). In this

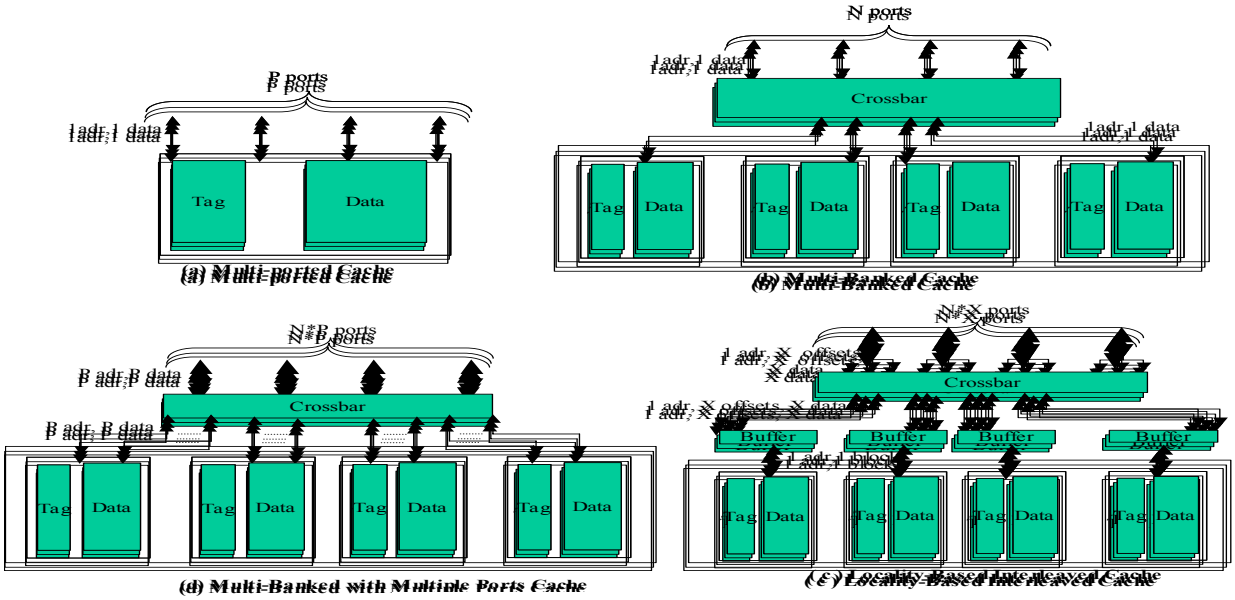

Figure 1: Multiported caches

paper, true multiported caches are used for comparison purposes only.

2.2. Multibanked cache

A multibanked cache, see Figure 1.b, consists of several cache banks. Each bank can service only one request per cycle since it has only one access port. A crossbar is needed to connect the various load/store units (LSUs) to the access ports of the cache.

2.2.1. Block-Interleaved cache (BI)

In a block-interleaved (BI) cache an entire cache line is placed in a bank, and the least significant bits in the cache index are used to determine the bank number[1]. In other words, consecutive cache lines are placed in consecutive banks. The BI scheme can support multiple requests per cycle when they do not map to the same bank. However, due to spatial locality in the data reference stream, multiple accesses to the same cache bank might occur simultaneously which cannot be supported by the BI cache.

2.2.2. Word-Interleaved cache (WI)

In a word-interleaved (WI) cache, some bits in the block offset are used to determine the bank number. This has the effect of spreading cache lines across banks. Consecutive addresses are mapped to different cache banks. Obviously, this is beneficial in case of spatial locality. Unfortunately, the WI scheme requires tag replication in each bank or multiporting the tag store to allow multiple accesses per cycle.

2.3. Hybrid multiported-multibanked caches

In this section, we present two hybrid multiported-multibanked cache designs, namely locality-based interleaved cache (LBIC) and a multibanked cache with multiple ports per bank (MBMP).

2.3.1. Locality-Based Interleaved Cache (LBIC)

Rivers et al. [5] propose the LBIC structure, see Figure 1.c, which extends the BI multibanked scheme by adding an X-ported single line buffer to each bank. This buffer combines multiple requests to the same cache block. As such, the cache bank is accessed only once for servicing multiple requests to the same cache block. An LBIC cache is represented by the couple (N,X) in which N is the number of banks and X is the number of ports per buffer. The value N·X represents the total number of cache ports.

2.3.2. Multibanked cache with multiple ports per bank (MBMP)

In a MBMP cache, see Figure 1.d, every cache bank has several independent cache ports. An MBMP cache is represented by the couple (N,P), where N is the number of banks and P is the number of ports per bank. Each bank can responds to P independent requests. The total number of

[1] A memory address here consist of a cache index and a block offset.

Table 1: Cacti parameter values for the different cache schemes.

	Block size B (in bytes)	No. of ports per bank P	No. of banks N	Data bus width DW (in bytes)	Address bus width AW (in bits)
BI	64	1	1 to 32	8	$32 - \log_2 N$
WI	8	1	1 to 32	8	$32 - \log_2 N$
LBIC	64	1	1 to 32	16 (X= 2)	$32 - \log_2 N + \log_2 B$
MBMP	64	2	1 to 16	8	$32 - \log_2 N$
PRF	64	1 to 32	1	8	32

ports in a MBMP cache is N·P. Like in the LBIC structure, several requests can be sent to the same bank but in contrast to the LBIC structure these requests must not obligatory concern the same block.

3. Cacti: an analytical model to evaluate cache structures

To evaluate the access time, the power consumption and the chip area of these multiported cache schemes, we have used an existing analytical cache model, namely Cacti [6]. Cacti v3.0 is an integrated access time, power consumption and chip area model for cache memories. Cacti supports multibanked caches where each bank has the same size and the same structure. The access time to a multibanked cache then is the access time to the bank most distant from the entry point of the cache (which yields the longest wires and thus the longest delay). Likewise we consider the energy consumed for accessing the most distant cache bank. Note that only the bank that is accessed is activated, i.e., the other banks are inactive or in other words, do not consume energy. In the Cacti model, each bank is composed of several units: A (with A the associativity of the cache) arrays for storing tags and data, A tag comparators and A multiplexers for selecting a word (typically 8 bytes) out of a cache line consisting of B bytes. In the experiments in this paper, the total cache size is fixed to 32KB for all possible cache configurations, the caches are two-way set-associative and a 0.1µm chip technology is assumed. Table 1 gives an overview of the various schemes used in this paper. The data bus width (DW) is the number of bytes that can be read from or written to the cache per access. The address bus width (AW) is the number of bits required to address data in the cache. The number of load/store units M is equal to M = N·P, with N the number of banks and P the number of ports per bank.

4. Crossbar for multibanked schemes

Figure 2 presents the architecture of the multibanked cache memory as well as the crossbar connecting the load/store units (LSUs) to the cache banks. We assume there are no conflicts between the different addresses generated by LSUs. In other words, collision detection and resolution logic is available in hardware, although not modeled here, to prevent different load/store instructions from accessing the same cache bank. Each demultiplexer in Figure 2 can service up to P requests to the same bank simultaneously.

4.1. The crossbar

As shown in Figure 3, the crossbar is composed of M demultiplexers. The longest distance in the crossbar between one LSU and the cache can be

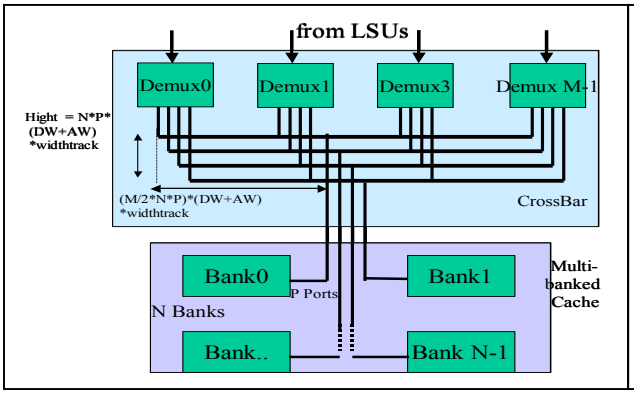

Figure 2: The cache structure and the crossbar

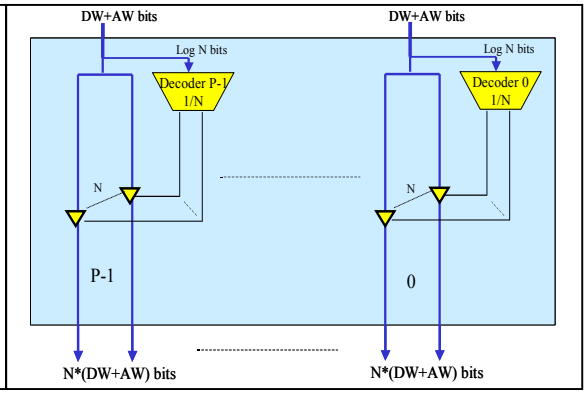

Figure 3: The demultiplexer structure

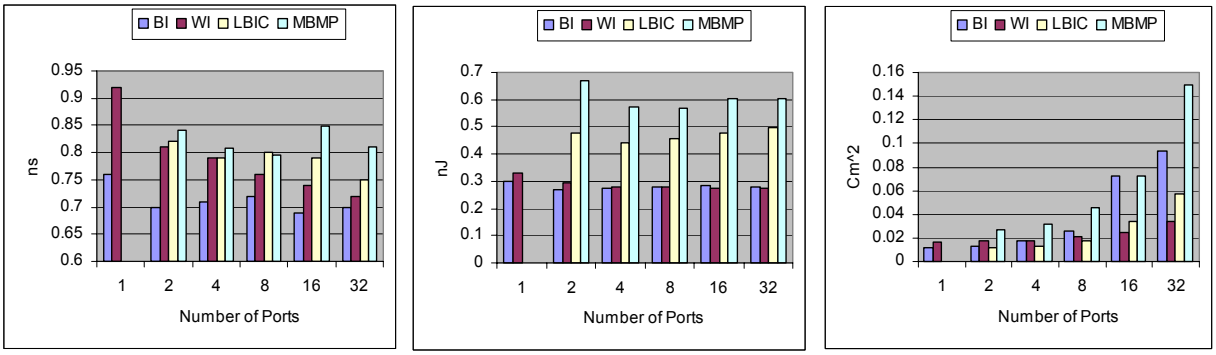

Figure 4: Access time, power consumption and chip area for various multiported caches (BI, WI, LBIC and MBMP) as a function of the number of ports (or the number of cache banks) without taking into account the crossbar.

calculated as follows: *crossbarWireLength = ½·(M·N·P)·(DW+AW)·width_track,* with *width_track* the width of a wire plus the space required between two adjacent wires. In this formula, we assume that the width of the crossbar is determined by the total number of wires coming out of the demultiplexers. Note that we only account for half the width since we assume that the LSUs are placed symmetrically to the entry point of the cache to minimize the access time to the cache, see Figure 2.

Analogeously, the height of the crossbar can be calculated: c*rossbarWireHeight = N·P·(DW+AW)·width_track.* To simplify the calculation, we do not take into account the wire length for connecting a LSU to a demultiplexor and the distance between the crossbar and the entry point of the cache memory. Moreover, these wire segments exist in all schemes (even in case of a true multiported cache).

As such,

- the total delay through the crossbar equals: *crossbarDelay(M,N,P) = wireDelay(M,N,P) + demuxDelay(N, P),*
- the power consumption of the crossbar for one access is : *crossbarPower(M,N,P) = wirePower(M,N,P) + demuxPower(N,P),* and
- the chip area occupied by the crossbar is: *crossArea(M,N,P) = wireArea(M,N,P) + M·demuxArea(N,P).*

WireDelay, wirePower and wireArea are the delay, the power consumption and the chip area of the wires, respectively. These parameters depend on the *crossbarWireLength* and the *crossbarWireHeight*; as such, they are a function of M, N, and P. In the experiments, they are measured using the models proposed in Cacti v3.0. DemuxDelay, demuxPower and demuxArea correspond to the delay, the power consumption and the chip area of the demultiplexer, respectively.

4.2. Delay, power consumption and chip area of the demultiplexer

To quantify the demultiplexer delay, the power consumption and the chip area of the demultiplexer, see Figure 3, we use decoders and tri-state drivers. For this purpose, we make use of validated models that are available in Cacti for these structures. Note that Figure 3 represents the general case where multiple independent access ports, say P, are available per cache bank. The experimental results showed that the chip area taken by demultiplexers is negligible in comparison to the total crossbar chip area. Therefore, only the delay and the power consumption of the demultiplexer are presented here: *demuxDelay(N,P) = decoderDelay(N) + driverDelay* and *demuxPower(N,P) = decoderPower(N) + driverPower.* In these formulae, decoderDelay, driverDelay, decoderPower and driverPower are computed using the analytical models provided within the Cacti framework. The length of the wires that connect the decoders to the tri-state drivers is calculated by the following equation: *decoderWire(N,P) = (N-1)·(AW+DW)·width_track.*

5. Experimental Results

In this section, we quantify and evaluate the access time, the power consumption and the chip area for the various multiported cache designs under study. First, we compare the various cache schemes without considering the crossbar. Second, we show how to optimize the LBIC structure. Third, we model the crossbar. Finally, we will compare the cache schemes while considering the crossbar.

5.1. Multibanked schemes without crossbar

Figure 4 shows the access time, the power consumption and the chip area for various multiported cache schemes (BI, WI, LBIC, MBMP). Note that, the access time and the power

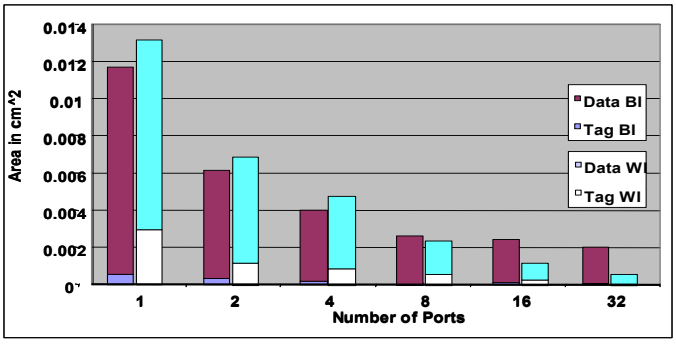

Figure 5: Occupied area by data and tag arrays in BI and WI inside one bank (the routing area is not considered).

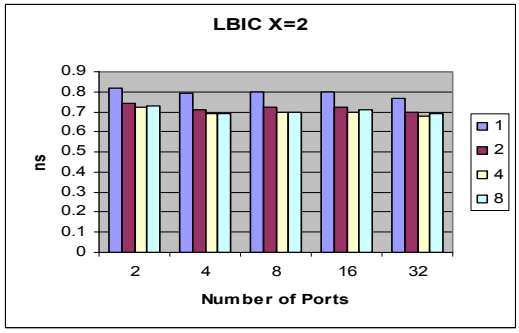

Figure 6: Access time for LBIC, BitOutDiv=1, 2, 4 and 8

consumption in this Figure are related to the most distant bank, and thus represent the maximum values. Note also that we do not take into account the crossbar so far; the crossbar will be considered in the following sections. For LBIC and MBMP, no results reported for one port per cache bank since these schemes correspond to the BI cache. Figure 4 shows that when the number of ports (or the number of cache banks) increases, the access time and power consumption remain constant or decrease slightly (for some schemes). The reason for this is that, for a fixed total cache size, the increase in the number of banks increases the address and data bit routing distance. Nevertheless, the access time and power consumption inside a bank decrease because of decreasing the bank size. In general, the access time for LBIC is larger than for BI and WI. In the next section, an optimization of LBIC will be presented. Note also that the MBMP scheme attains the largest access time for a large number of cache ports, e.g., 18% larger than in case of BI.

The middle graph of Figure 4 shows the power consumption to access the most distant bank . BI, WI and MBMP consume more or less the same amount of energy per access. Because of the large word size of LBIC, the power consumption of this scheme is larger than for WI and BI. The right graph of Figure 4 presents the chip area occupied by the four schemes. When the number of ports is smaller than 8, LBIC needs the smallest area. Under the same conditions, BI occupies less area than WI. This is due to the chip area required for tag replication. Nevertheless, for more than 8 cache ports, WI occupies less area than BI. This surprising result can be explained by the fact that when the bank size is reduced, the data array occupies a relatively important area in BI, while at the same time, the tag array in WI occupies relatively less area because there are less sets, see Figure 5.

5.2. Optimizing LBIC

In the previous section we have shown that LBIC attains access times that are higher than the BI and WI access times. The reason for this is the fact that the output multiplexer in Cacti needs to drive a large number of bits (1 block), multiple words instead of a single word in other schemes (see Figure 35 in [9]). As such, we propose to reduce the delay of the output multiplexer in the LBIC scheme. This optimization consists in reducing the number of output drivers connected to a single control signal. By multiplying the number of control signals by a factor *BitOutDiv* and dividing the number of output drivers per control signal by the same factor *BitOutDiv*, the total delay of the output multiplexer can be reduced. Figure 6 shows the access time reduction with our optimization in LBIC in case we assume 2 words accessed in parallel (X=2). Our results show a significant reduction in access time. For the remainder of this paper, we will use the LBIC configuration with *BitOutDiv=2*. Moreover, this optimization adds a negligible power consumption because, the multiplexor drivers consume a small percentage of the total power (less than 4% of the total energy) [4].

5.3. Delay, power consumption and chip due to crossbar

Figure 7 presents the delay, the power consumption as well as the chip area due to a crossbar with an 8 bytes data bus and a 32 bits address bus. This configuration corresponds to a crossbar used in WI, BI and MBMP and will be further referred to as "XbarWI/BI/MBMP". For large numbers of cache ports, wires are the dominant component in the crossbar. Compared to XbarWI/BI/MBMP, the crossbar for LBIC "XbarLBIC" uses fewer buses, see Figure 1. As a result, XbarLBIC is faster, consumes less energy and needs less chip area than XbarWI/BI/MBMP, see Figure 8.

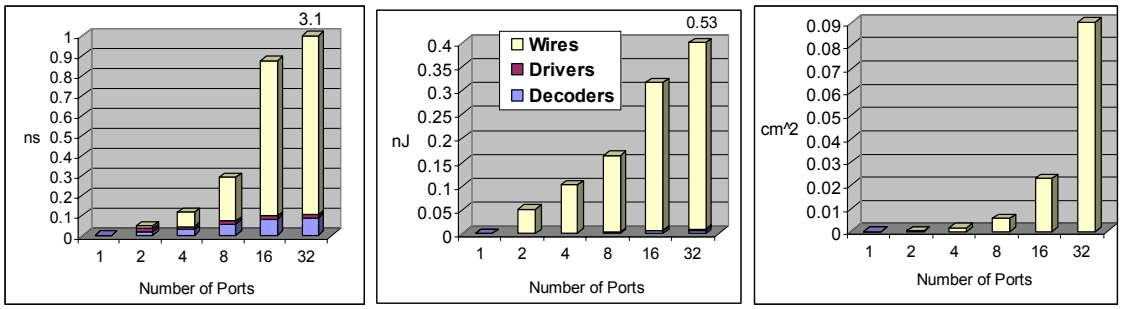

Figure 7: Delay, power consumption and area due to a crossbar with one 8-byte data bus and one 32-bit address bus.

5.4. Comparing the different schemes with crossbar

The access time, the power consumption per access and the chip area (including the crossbar for multiported caches) are given in Figure 8 (A to C) together with the additional crossbar overheads with XbarWI/BI/MBMP and XbarLBIC. From these graphs, we can see that the access time for BI is smaller than for WI and MBMP. The access time for LBIC is even smaller than for BI. Note also that MBMP is significantly smaller than the true multiported cache.

6. Related work

A large amount of work has been devoted to high performance data cache design. In this section, we only discuss the contributions most closely related to the work presented in this paper.

Cho et al. [1] propose data decoupling as a solution to reduce the complexity of multiported caches. Their solution consists in using two distinct memory pipelines and caches each having few access ports. The memory stream is divided in several independent streams (data stream, heap stream, and stack stream) by using access region prediction. The various memory streams are then directed to distinct memory pipelines.

Juan et al. [2] evaluate several multiported data caches. After presenting a classification of the different schemes, they compare their performances in terms of IPC and occupied chip area. From their experiments, they conclude that multibanked cache schemes attain better performance compared to a true multiported cache at a reduced hardware cost. Their measurement of hardware complexity is based only on the chip area occupied by the cache banks. Our work supplements this study and offers a more accurate comparison. This is done by taking into account the crossbar and by evaluating each scheme in terms of access time, power consumption and chip area.

Rivers et al. [5] explore a well known memory reference stream characteristic, namely the spatial

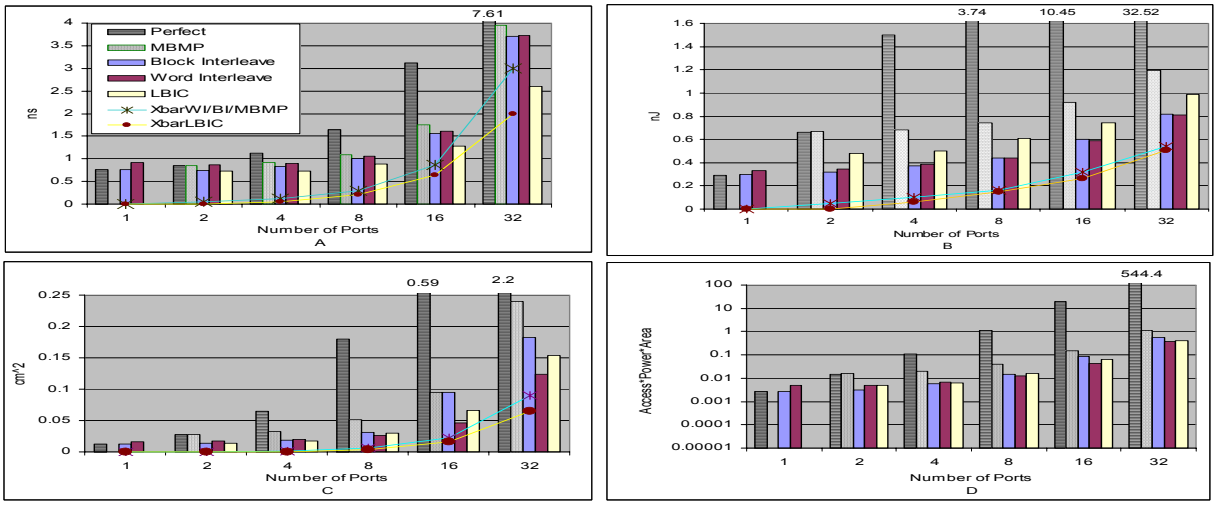

Figure 8: Access time, power consumption and chip area for multiported caches as a function of the number of cache ports (or the number of cache banks) while taking into account the crossbar.

locality within a single cache line. For this purpose, they add a multiported single line buffer to each cache bank. The cache design they propose is called Locality-Based Interleaved Cache (LBIC) and was used in this paper as one of the multiported cache design alternatives.

Recently, Vandierendonck and De Bosschere [8] proposed a new multibanked interleaving scheme called "transformed block interleaving (TFBI)". This interleaving scheme is based on BI. But before accessing the cache, the n bits specifying the bank number are swapped with the n most significant bits in the block offset field. As such, they try to combine the benefits of block-interleaving (low hardware cost) and word-interleaving (high IPC). In other words, the performance in terms of IPC is increased compared to the block-interleaved design without increasing its hardware complexity.

7. Conclusion

As current and future microprocessors implement bigger instruction window and wider issue widths, an increasing number of load/store instructions need to be executed per clock cycle to be able to deliver a high performance. As such, caches need to be multiported. Since true multiporting is too expensive in terms of access time, power consumption and chip area, other solutions need to be proposed. In the literature, the notion of multibanked caches is well known. Indeed, several researchers have investigated the performance of multibanked schemes and have proposed improvements on the basic concepts of multibanked caches. These studies mainly focused on performance in terms of IPC. In this paper, we quantify and evaluate the performance of such cache designs in terms of access time, power consumption and chip area. This research can thus be viewed as a useful complement to existing work in the field of multiported caches.

For this study, we extended and used Cacti, an analytical cache model capable of estimating access time, power consumption and chip area accurately. The main issue that needed to be addressed in this study was modeling the crossbar that connects the load/store units to the cache ports. As such, we were able to evaluate five possible multiported cache designs: true multiported caches, block-interleaved multibanked caches, word-interleaved multibanked caches, locality-based interleaved caches (LBIC) and multibanked caches with multiple port per bank (MBMP).

From the experiments conducted in this paper, we can conclude that

- the access time to a multibanked cache (without considering the crossbar) is relatively independent of the number of cache banks;
- when taking the crossbar into account, the access time of the cache is highly affected by the crossbar delay;
- true multiporting is much more expensive in terms of access time, power consumption and chip area than the other multiported cache schemes;
- the delay, the energy consumption and the chip area of the crossbar is mainly due to wiring;
- the access time of the locality-based interleaved cache (LBIC) can be significantly reduced by reducing the delay of the output multiplexer;
- the LBIC scheme performs better than the other schemes investigated in terms of access time and energy consumption; word-interleaving results in the smallest chip area for a large number of banks.

References

[1] S. Cho, P. C. Yew and G. Lee. *A high bandwidth memory pipeline for a wide issue processors*, IEEE Transactions on Computers, Vol. 50, No.7, July 2001, p.706-723.

[2] T. Juan, J. J. Navarro and O. Temam. *Data caches for superscalar processors.* Proceedings of the 11th International Conference on Supercomputing, July 1997, p. 60-67.

[3] R.E. Kessler. *The Alpha 21264 Microprocessor.* IEEE Micro, March-April 1999, p. 24-36.

[4] G. Reinman and N. Jouppi. Cacti 2.0: *An Integrated Cache Timing and Power Model,* Research Report Western Research Lab 1999.

[5] J. A. Rivers, G. S. Tyson, E. S. Davidson, T. M. Austin. *On high-bandwidth data cache design for multi-issue processors*. Proceedings of 30th ACM/IEEE International Symposium on Microarchitecture (MICRO-30), 1997, p. 46-56.

[6] P. Shivakumar and N. P. Jouppi. *CACTI 3.0: An integrated cache timing, power, and area model*, Research Report Western Research Lab 2001/2, August 2001.

[7] G. S. Sohi and M. Franklin. *High-bandwidth data memory systems for superscalar processors.* Proceedings of the fourth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-IV), April 1991, p. 53-62.

[8] H. Vandierendonck and K. De Bosschere. *An address transformation combining block and word interleaving*. Computer Architecture Letters, Volume 1, June 2002.

[9] S. Wilton and N. Jouppi. *An Enhanced Access and Cycle Time Model for On-Chip Caches.* Research Report Western Research Lab, 93/5, July 1994.

[10] W. Wulf and S. McKee. *Hitting the Memory Wall: Implications of the Obvious*. Computer Architecture News, 23(1):20–24, March 1995.

[11] K. C. Yeager. *The MIPS R10000 superscalar microprocessor*. IEEE Micro, March-April 1996.