Bank-interleaved cache or memory indexing does not require euclidean division

André Seznec

Bank-interleaved cache or memory indexing does not require euclidean division

André Seznec, IRISA/INRIA, Campus de Beaulieu
35042 Rennes Cedex, FRANCE
e-mail : andre.seznec@inria.fr

Abstract

Concurrent access to bank-interleaved memory structure have been studied for decades, particularly in the context of vector supercomputer systems. It is still common belief that using a number of banks different from 2^n leads to insert a complex hardware including a non-trivial divider on the access path to the memory.

In 1993, two independent studies [1], [2] were showing that through leveraging a very simple arithmetic result, the Chinese Remainder Theorem, this euclidean division is not needed when the number of banks is prime or simply odd. In the mid 90's, the interest for vector supercomputers faded and the research topic disappeared. The interest for bank-interleaved cache has reappeared recently [3] in the GPU context.

In this short paper, we extend the result from [1] and we show that, regardless the number of banks:

Bank-interleaved cache or memory indexing does not require euclidean division.

1 Introduction

The need for concurrent access to data in memory structures has lead to the design and use of banked-interleaved structures, first for main memory, e.g. in vector supercomputers and at second step in caches. Optimizing parallel access to this bank-interleaved memory structure has been studied for decades particularly in the 80's and the early 90's in the context of the access to strided vector processors [4], [5], [6], [7], [8], [9]. More recently, similar studies have been published in the context of bank-interleaved caches for vector processors [10], [11] or GPU caches [3].

It was common belief that for simple indexing of a bank-interleaved cache or memory, the number of banks should be a power of 2, 2^n, since otherwise complex arithmetic including an euclidean division and euclidean modulo would be required [12], [6]. Two independent studies published in 1993 at the ISCA conference [1], [2] pointed out that for prime or odd numbers of banks such division is useless due a very simple arithmetic theorem, the Chinese Remainder Theorem.

In a recent study, Diamond et al. [3] point out that, in the context of GPU, the best number of cache banks might not be a power of two but may be any other number. They propose an optimized hardware mechanism to compute both modulo and division at the same time. While their hardware proposal optimizes the implementation of bank-interleaved cache or memory indexing if the euclidean division was required, such an euclidean division is useless.

In this short paper, we extend the result from [1], [2] showing that the euclidean division is useless for indexing a bank-interleaved memory or cache regardless the number of banks.

Notation and definition

For convenience, in the remainder of the paper, we will refer only to a bank-interleaved memory. The result in the paper also applies to bank-interleaved caches.

We will refer to a bank-interleaved memory with an odd number of banks as an odd memory system.

Bank interleaving in a memory or cache can be implemented at different granularities; for instance 8-byte word on Cray vector supercomputers, cache blocks on vector microprocessor [11], 8-byte word for recent GPUs. For convenience, we will refer to this granularity as word in the remainder of the paper. All the addresses that will be considered in this paper will be in words, therefore ignoring offset in the word.

2 Odd memory systems do not require Euclidean division

This section summarizes the results published in [1].

2.1 Usual data mapping in a bank-interleaved memory

The physical mapping of word at address A, $0 \leq A < 2^c * N$ on memory is defined by its bank number $m(A)$, $0 \leq m(A) < N$, and its local address $l(A)$, $0 \leq l(A) < 2^c$, in the bank.

The most important property that most manufacturers want to guarantee when using in a N-bank interleaved memory is the parallel access to consecutive words in memory, i.e. any N consecutive words are stored. This leads to the conventional mapping of data defined by:

- $m(A) = A \ mod \ N$

Bank number

Local address	0	1	2	3	4
0	0	1	2	3	4
1	5	6	7	8	9
2	10	11	12	13	14
3	15	16	17	18	19

Fig. 1. *mod-div* mapping on a 5-bank interleaved memory

- $l(A) = A \ / \ N$

We will refer to this address mapping as the *mod-div* mapping. For N=2^n, the *mod-div* mapping, the bank number consists in the n least significant bits while the high order bits constitute the local address. Figure 1 illustrates this mapping for N=5 and a memory bank of 4 words.

The old case for prime memory system

Using prime (or odd) memory system has been known to provide interesting properties for vector supercomputers since 1971. Budnick [4] established the following property on distribution of the elements among the memory banks.

Theorem 2.1 (Distribution Theorem). When the bank distribution function is defined by $m(A) = A \ mod \ N$, then for any vector V stored with a stride R, $V(i)$ and $V(j)$ are stored in the same memory bank iff $i = j$ mod $N/GCD(N, R)$

Then for any vector V stored with a stride R, $N/GCD(N, R)$ consecutive elements of the vector are stored in distinct memory banks.Using a prime number of memory banks ensures a conflict free distribution of any slice of N consecutive elements for all the vectors stored with a stride R not multiple of N. Moreover, using a prime number induces simple control for memory accesses; only two distributions of elements of a vector slice are possible: conflict free access is possible *or* all the elements lie in the same memory bank. A last argument in favor of using a prime memory system is the demand for memory throughput on vector accesses with power-of-two strides in some specific applications.

What was (believed to be) wrong with prime memory systems

Unfortunately when using the usual data mapping, address computation for a prime memory system requires arithmetic modulo a fixed prime number:

1) Computing the memory bank number for word at address A requires the computation of $A \ mod \ N$. For specific values, very fast hardware evaluations of such a modulo may be implemented.
2) The computation of the local displacement in the memory bank requires an Euclidean Division by N and this division is quite complex when N is an odd number. This Euclidean Division may lenghten significantly the total memory indexing.

Therefore, when the usual low-order mapping on memory is used in a vector machine, the number of memory banks was a power of two.

In fact by changing the choice of the local address function, this Euclidean Division can be avoided on memory systems with an odd number of banks, but the result [1], [2] was probably published too late (1993) to be used in vector supercomputers.

2.2 Simple is better

A very old arithmetic result known as Chinese Remainder Theorem [1] induces a very elegant way to map elements onto a parallel memory consisting in an odd number N banks of 2^c elements and for which no hardware is needed to compute the local address[2].

Theorem 2.2 (Chinese Remainder Theorem). Let P and Q be 2 relatively prime integers, i.e. GCD(P,Q)=1 then
for each pair (X, Y) such that $0 \leq X < P$ and $0 \leq Y < Q$
there exists one **and** only one $0 \leq Z < P * Q$ such that : $Z \equiv X \ mod \ P$ and $Z \equiv Y \ mod \ Q$

The Chinese Remainder Theorem just guarantees that, 2^c being the number of memory words per bank, the functions

1. It seems that this result was known more than 2000 years ago by the old Chinese
2. On the Burroughs Scientific Processor [12], the Euclidean Division was also avoided, but $\frac{1}{17} \ th$ of the memory was wasted.

Bank number

Local address	0	1	2	3	4
0	0	16	12	8	4
1	5	1	17	13	9
2	10	6	2	18	14
3	15	11	7	3	19

Fig. 2. *mod-mod* mapping on a 5-bank interleaved memory

defined by $m(A) = A \ mod \ N$ and $l(A) = A \ mod \ 2^c$ define a mapping of the address space onto the physical memory since N is odd and therefore prime with 2^c. This *mod-mod* mapping is illustrated in Figure 2.

The bank number function is exactly the same as for the *mod-div* mapping. Therefore the Distribution Theorem still holds for this mapping: conflict free access is possible to any slice of N consecutive elements of a vector stored with a stride R not multiple of N.

The main benefit of this *mod-mod* mapping is that the local address $l(A)$ is the c least significant bits of the address: no hardware is required for deriving it from the address.

Then we can state:

"Odd Memory Systems Do Not Require Euclidean Division"

3 No bank-interleaved memory system require euclidean division

In their experiments, Diamond et al [3] points out that for GPUs powers of two are not the best number of banks for a GPU cache. In their particular experiments, they argue to use 62 and 48 cache banks. These numbers are neither odd nor power of two. However, in this section we extend the result from the previous section to every number N of banks.

We consider the general form of an integer as N= 2^n R with R odd. The particular cases of N being a power of two (R=1) and N being odd ($n = 0$) have been treated in the previous section. Therefore, we assume $n \neq 0$ and R odd, but greater than 1.

We consider the two functions m and l defined by:

- $m(A) = (A \ mod \ N)$
- $l(A) = \frac{A}{2^n} \ mod \ 2^c$

These functions define a mapping from the address space to the physical memory as shown below:

$A = (A \ mod \ 2^n) + 2^n * \frac{A}{2^n}$, therefore $m(A) = (A \ mod \ 2^n) + 2^n * (\frac{A}{2^n} \ mod \ R)$.

The application of the Chinese Remainder Theorem ensures that the functions l and $\frac{m}{2^n}$ define is a one-to-one mapping from $\{0, .., R*2^c-1\}$ onto $\{0, .., 2^c-1\}*\{0, .., R-1\}$. As a consequence, the functions l and m define is a one-to-one mapping from the address space $\{0, .., N * 2^c - 1\}$ onto the set of memory words of the memory system. This memory mapping is illustrated for a 6-bank interleaved memory on Figure 3.

Therefore we can state :

no euclidean division is needed to index a bank-interleaved memory system.

4 Bank number computation

The computation of modulo P is simple for P=$2^p - 1$ or P= $2^p + 1$, as well as N= $2^c * (2^p - 1)$ or N=$2^c * (2^p - 1)$. This can implemented very simply through cascaded carry save adders followed by last p bits adder and very limited logic.

For example, a GPU with 32 warps would feature 32 to 64 cache banks. In that range, many numbers are of the form N= $2^c * (2^p - 1)$ or N =$2^c * (2^p + 1)$: 33, 34, 36, 40, 48, 56, 60, 62 and 63.

5 Conclusion

Despite publications in 1993 [1], [2], it is still common belief that indexing a bank interleaved memory or cache requires an euclidean division when the number of banks is not a power of two. In [1], [2], it was shown that euclidean division is not required for prime or odd numbers of banks.

In this short paper, we have trivially extended this result to any number of banks. Therefore, regardless the number of banks:

Bank-interleaved cache or memory indexing does not require euclidean division.

Bank number

Local address	0	1	2	3	4	5
0	0	1	8	9	16	17
1	18	19	2	3	10	11
2	12	13	20	21	4	5
3	6	7	14	15	22	23

Fig. 3. Euclidean division free mapping on a 6-bank interleaved memory

ACKNOWLEDGEMENT

This work was partially supported by the European Research Council Advanced Grant DAL No. 267175.

REFERENCES

[1] A. Seznec and J. Lenfant, "Odd memory systems may be quite interesting," in *Proceedings of the 20th Annual International Symposium on Computer Architecture. San Diego, CA, May 1993*, 1993, pp. 341–350. [Online]. Available: http://doi.acm.org/10.1145/165123.165175

[2] Q. S. Gao, "The chinese remainder theorem and the prime memory system," in *Proceedings of the 20th Annual International Symposium on Computer Architecture*, ser. ISCA '93. New York, NY, USA: ACM, 1993, pp. 337–340. [Online]. Available: http://doi.acm.org/10.1145/165123.165172

[3] J. Diamond, D. Fussell, and S. W. Keckler, "Arbitrary modulus indexing," in *Proceedings of the 47th ACM/IEEE symposium on Microarchitecture*, Dec 2014.

[4] P. Budnik and D. J. Kuck, "The organization and use of parallel memories," *IEEE Trans. Comput.*, vol. 20, no. 12, pp. 1566–1569, Dec. 1971. [Online]. Available: http://dx.doi.org/10.1109/T-C.1971.223171

[5] D. T. Harper, III and J. R. Jump, "Vector access performance in parallel memories using skewed storage scheme," *IEEE Trans. Comput.*, vol. 36, no. 12, pp. 1440–1449, Dec. 1987. [Online]. Available: http://dx.doi.org/10.1109/TC.1987.5009496

[6] B. R. Rau, "Pseudo-randomly interleaved memory," in *Proceedings of the 18th Annual International Symposium on Computer Architecture*, ser. ISCA '91. New York, NY, USA: ACM, 1991, pp. 74–83. [Online]. Available: http://doi.acm.org/10.1145/115952.115961

[7] A. Seznec and J. Lenfant, "Interleaved parallel schemes: Improving memory throughput on supercomputers," in *Proceedings of the 19th Annual International Symposium on Computer Architecture*, ser. ISCA '92. New York, NY, USA: ACM, 1992, pp. 246–255. [Online]. Available: http://doi.acm.org/10.1145/139669.140381

[8] M. Valero, T. Lang, and E. Ayguadé, "Conflict-free access of vectors with power-of-two strides," in *ICS*, 1992, pp. 149–156. [Online]. Available: http://doi.acm.org/10.1145/143369.143403

[9] B. D. de Dinechin, "A ultra fast euclidean division algorithm for prime memory systems," in *Proceedings of the 1991 ACM/IEEE Conference on Supercomputing*, ser. Supercomputing '91. New York, NY, USA: ACM, 1991, pp. 56–65. [Online]. Available: http://doi.acm.org/10.1145/125826.125874

[10] A. Seznec and R. Espasa, "Conflict-free accesses to strided vectors on a banked cache," *IEEE Trans. Computers*, vol. 54, no. 7, pp. 913–196, 2005. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/TC.2005.110

[11] R. Espasa, F. Ardanaz, J. Gago, R. Gramunt, I. Hernandez, T. Juan, J. S. Emer, S. Felix, P. G. Lowney, M. Mattina, and A. Seznec, "Tarantula: A vector extension to the alpha architecture," in *29th International Symposium on Computer Architecture (ISCA 2002), 25-29 May 2002, Anchorage, AK, USA*, 2002, p. 281. [Online]. Available: http://dx.doi.org/10.1109/ISCA.2002.1003586

[12] D. H. Lawrie and C. R. Vora, "The prime memory system for array access," *IEEE Trans. Comput.*, vol. 31, no. 5, pp. 435–442, May 1982. [Online]. Available: http://dx.doi.org/10.1109/TC.1982.1676020