UNITÉ DE RECHERCHE
INRIA-RENNES

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tél.:(1) 39 63 55 11

Rapports de Recherche

1992

25ème anniversaire

N° 1654

Programme 1
Architectures parallèles, Bases de données,
Réseaux et Systèmes distribués

PRIME MEMORY SYSTEMS DO NOT REQUIRE EUCLIDEAN DIVISION BY A PRIME NUMBER

André SEZNEC
Yvon JÉGOU
Jacques LENFANT

Avril 1992

RR-1654

IRISA

INSTITUT DE RECHERCHE EN INFORMATIQUE
ET SYSTEMES ALEATOIRES

Campus Universitaire de Beaulieu
35042 - RENNES CEDEX FRANCE
Tél. : 99 84 71 00 - Télex : UNIRISA 950 473 F
Télécopie : 99 38 38 32

Prime Memory Systems Do Not Require Euclidean Division By a Prime Number

André Seznec, Yvon Jégou, Jacques Lenfant
Programme 1
Projet CALCPAR

20 mars 1992
Publication Interne n° 644, Mars 1992, 10 pages

Comment utiliser un nombre premier de bancs sans savoir faire une division euclidienne

Résumé

L'utilisation d'un nombre premier N de bancs pour construire la mémoire d'un processeur vectoriel permet un accès sans conflit à toute tranche de N éléments consécutifs d'un vecteur rangé avec une raison d'accès non multiple de N.

Il est communément admis que l'utilisation d'un tel nombre premier N de bancs entraine l'utilisation d'une division euclidienne par N lors du calcul de chaque adresse. Dans cette note, nous montrons que le théorème Chinois permet de définir un rangement des données dans la mémoire qui permet d'éviter cette division euclidienne.

Abstract

Using a prime number N of memory banks on a vector processor allows a conflict-free access for any slice of N consecutive elements of a vector stored with a stride not multiple of N.

To reject the use of such a prime number of memory banks, it is generally advanced that address computation for such a memory system would require systematic Euclidean Division by the prime number N. In this short note, we show that there exists a very simple mapping of data in the memory banks for which address computations does not require any Euclidean Division.

1 Introduction

As vector parallelism is easiest to detect and to exploit, a large number of vector processors have been built and commercialized since 1965. Vector processors usually resort to the pipeline architecture (e.g., vector register machines as the Cray Series) or to the SIMD architecture where a single sequencing unit controls several identical processing elements (e.g. the Connection Machine).

Most of the manufacturers of vector processors consider that a vector is an ordered set of words whose addresses form an arithmetic series.

In most cases, the bottleneck for performance on vector applications is the parallel (or pipelined) access to vectors in memory: conflicts may arise when accessing vectors. Since 1970, a lot of results have been presented [1, 3, 5, 6, 9, 10] in order to partially remove this bottleneck.

Using a prime number N of memory banks in the memory of a vector processor allows a conflict-free access for any slice of N consecutive elements of a constant-strided vectors which stride is not multiple of N [1]. We shall refer to a memory system built with a prime number of memory banks as a prime memory system [8].

To reject the use of prime memory systems, it is generally advanced that address computation requires an Euclidean Division by the prime number N or that some part of the memory space has to be wasted [8]. In this note, we present a very simple mapping of data on a prime memory system for which address computation does not require Euclidean Division and which does not waste any part of the memory space.

2 Usual mapping data in a parallel memory

In both SIMD machines and vector register processors, several memory banks are used in the design of the memory system.

To obtain a maximum memory throughput on a vector instruction, conflicts on accesses to memory must be avoided.

The mapping of an address A on memory is defined by two functions:

- $m(A)$ the number of the memory bank where word at address A is mapped. We shall refer to m as the bank distribution function.
- $l(A)$ the local address in memory bank $m(A)$. We shall refer to l, as the local displacement.

The usual mapping function used in most of the current vector processors is defined by:
let N be the number of memory banks

- $m(a) = A \ mod \ N$
- $l(A) = A \ / \ N$

This mapping is sometimes referred to as low order interleaving.

Figure 1 illustrates this mapping for N=13 and a memory bank size of 16 words.

The following property on distribution of the elements among the memory banks is known since 1971 [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 $i = j \bmod N/GCD(N, R)$

****

Figure 1: Usual mapping in a 13 banks parallel memory

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	1	2	3	4	5	6	7	8	9	10	11	12
1	13	14	15	16	17	18	19	20	21	22	23	24	25
2	26	27	28	29	30	31	32	33	34	35	36	37	38
3	39	40	41	42	43	44	45	46	47	48	49	50	51
4	52	53	54	55	56	57	58	59	60	61	62	63	64
5	65	66	67	68	69	70	71	72	73	74	75	76	77
6	78	79	80	81	82	83	84	85	86	87	88	89	90
7	91	92	93	94	95	96	97	98	99	100	101	102	103
8	104	105	106	107	108	109	110	111	112	113	114	115	116
9	117	118	119	120	121	122	123	124	125	126	127	128	129
10	130	131	132	133	134	135	136	137	138	139	140	141	142
11	143	144	145	146	147	148	149	150	151	152	153	154	155
12	156	157	158	159	160	161	162	163	164	165	166	167	168
13	169	170	171	172	173	174	175	176	177	178	179	180	181
14	182	183	184	185	186	187	188	189	190	191	192	193	194
15	195	196	197	198	199	200	201	202	203	204	205	206	207

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 and then may be accessed in parallel. Using a prime number of memory banks ensures conflict free parallel accesses to all 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.

In statistics on vector applications, members of the Fujitsu VP200 design team [11] note that about 80 % of the vectors are accessed with stride one, and the others are accessed with approximately randomly distributed strides. In order to illustrate the benefit that may be obtained using a prime number instead of a classical power of two as number of memory banks, we consider the average memory throughput on vector accesses of two memory systems : a 257 banks memory system and a 256 banks memory system.

> Let us consider a SIMD vector processor consisting in N processors connected to a parallel memory consisting in N memory banks and that a memory request to a vector slice of N consecutive elements may be issued on each cycle.
>
> Let us also consider that each bank is busy for one cycle by a memory request.
>
> When considering the distribution of vector strides previously referred, accessing 100 vector slices of N consecutive elements will require an average of 120 memory cycles (i.e $80 + 20 * (256/257) + 20 * (1/257) * 257)$) on a 257 banks memory system against an average of $180 = 80 + 20 * \Sigma_{i=0,7} * 2^i/2^{i+1} + 20 * (1/256) * 256$ memory cycles on a 256 banks memory system.

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 specialized applications [9].

What is 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 requires the computation of a modulo $A \ mod \ N$: for arbitrary values of N, there are no very efficient general technique to compute this modulo in hardware. But for particular values of N (for example if there exists s such that $N = 2^s - 1$ or $N = 2^s + 1$, or if there

exists s and relatively small x such that $x * N = 2^s - 1$ or $x * N = 2^s + 1$), simple and fast VLSI implementations can be derived from very simple mathematical properties:

As an example we consider the particular case of N= 13 and an address space smaller than 2^{30} words:

let us consider an address $A = \Sigma_{i=0,4} A_i 2^{6*i}$ with $0 \leq A_i < 64$

As $65 = 5 * 13$, $A \ mod \ 13 = (A_0 - A_1 + A_2 - A_3 + A_4) \ mod \ 13$,

A quite simple VLSI implementation of the computation of the remainder of a division by 13 can be easily be derived from this formula.

Adding such a piece of hardware in the final stage of an address generator does not significantly increase the total memory access time on a time multiplexed memory system.

2. Computing the local address in the memory bank requires an Euclidean Division by N is quite complex when N is an odd number (in [2], a complex hardware mechanism for fast division by $2^s - 1$ or $2^s + 1$ is presented).

Therefore, all vector machines which uses the usual low-order mapping on memory have been built a power of two as number of memory banks.

In the next sections, we shall see that by only changing the choice of the local displacement function, the Euclidean Division by a prime number can be avoided on prime memory systems.

3 The BSP-like mapping

It seems that there has only been one vector machine built with a prime memory system: the Burroughs Scientific Processor (BSP) [4, 8].

The memory system of the BSP is described in [8]. N= 17 memory banks were used.

The BSP was a SIMD vector machine with 16 pipelined processing elements. Pipelined were very short (in terms of cycles number). Adding an Euclidean division in address computation would have severely increased the number of stages in the pipeline for accessing to the memory.

The designers of BSP chose the slightly different mapping of data in memory described below:

Let $P \leq N$, let 2^c be the size of one memory bank:

- $m(A) = A \ mod \ N$
- $l(A) = P \ / \ N$

As $P \leq N$, these two functions define a mapping from the address space $\{0, .., P * 2^c - 1\}$ to the set of memory words in the memory system.

P may be chosen such as the euclidean division by P is easy: a power of two. In the BSP, P=16. This mapping is illustrated for N=13 and P=8 in figure 2. xx represents not mapped locations in memory.

As the local displacement does not influence the distribution of words in the memory banks, the Distribution Theorem still holds for this mapping: conflict free access is possible to any slices of N consecutive elements of a vector stored with a stride R not multiple of N.

The major draw back of this mapping is to waste part of the memory: on the N words having the same local address X, N-P words cannot be addressed. This was quite acceptable for the BSP with N=17 and P=16, but would not be acceptable for others values such as N=127 and P= 64.

4 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:

[1] It seems that this result was known more than 2000 years ago by the old Chinese

Figure 2: The BSP data mapping

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	1	2	3	4	5	6	7	xx	xx	xx	xx	xx
1	13	14	15	xx	xx	xx	xx	xx	8	9	10	11	12
2	xx	xx	xx	16	17	18	19	20	21	22	23	xx	xx
3	26	27	28	29	30	31	xx	xx	xx	xx	xx	24	25
4	39	xx	xx	xx	xx	xx	32	33	34	35	36	37	38
5	xx	40	41	42	43	44	45	46	47	xx	xx	xx	xx
6	52	53	54	55	xx	xx	xx	xx	xx	48	49	50	51
7	xx	xx	xx	xx	56	57	58	59	60	61	62	63	xx
8	65	66	67	68	69	70	71	xx	xx	xx	xx	xx	64
9	78	79	xx	xx	xx	xx	xx	72	73	74	75	76	77
10	xx	xx	80	81	82	83	84	85	86	87	xx	xx	xx
11	91	92	93	94	95	xx	xx	xx	xx	xx	88	89	90
12	xx	xx	xx	xx	xx	96	97	98	99	100	101	102	103
13	104	105	106	107	108	109	110	111	xx	xx	xx	xx	xx
14	117	118	119	xx	xx	xx	xx	xx	112	113	114	115	116
15	xx	xx	xx	120	121	122	123	124	125	126	127	xx	xx

- No hardware is needed to deduce the local address
- The whole address space may be used

Chinese Remainder Theorem:
Let p,q two relatively prime integers, then
for each $0 \leq a < p$, $0 \leq b < q$, there exists one and only one $0 \leq c < p * q$ such that $c \ mod \ p = a$ and $c \ mod \ q = b$.

****

Proof of this theorem is immediate by absurdity.

The Chinese Remainder Theorem ensures that the function defined below from the address space $\{0, .., N * 2^c - 1\}$ to the the set of memory words of the memory system is a one-to-one mapping:

- $m(A) = A \ mod \ N$
- $l(A) = A \ mod \ 2^c$

Implementation of the local displacement function requires no hardware: the c least significant bits of the address are directly used.

As in the previous section, the bank distribution function is the same as for the usual low order mapping, 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.

5 Conclusion

Using a prime memory system for a vector processor allows a conflict-free access for any slice of N consecutive elements of a constant-strided vector which stride is not multiple of N. As shown in section 2, this can significantly improve the average memory throughput on vector applications.

It was currently admitted in the computer architecture community that using an odd number N of memory banks induces Euclidean Division by N for computing the local displacement in the memory bank or induces the loss of some part of the memory as in the Burroughs Scientific Processor. As very fast Euclidean Division

Figure 3: Mapping function induced by Chinese Remainder theorem: N=13

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	144	80	16	160	96	32	176	112	48	192	128	64
1	65	1	145	81	17	161	97	33	177	113	49	193	129
2	130	66	2	146	82	18	162	98	34	178	114	50	194
3	195	131	67	3	147	83	19	163	99	35	179	115	51
4	52	196	132	68	4	148	84	20	164	100	36	180	116
5	117	53	197	133	69	5	149	85	21	165	101	37	181
6	182	118	54	198	134	70	6	150	86	22	166	102	38
7	39	183	119	55	199	135	71	7	151	87	23	167	103
8	104	40	184	120	56	200	136	72	8	152	88	24	168
9	169	105	41	185	121	57	201	137	73	9	153	89	25
10	26	170	106	42	186	122	58	202	138	74	10	154	90
11	91	27	171	107	43	187	123	59	203	139	75	11	155
12	156	92	28	172	108	44	188	124	60	204	140	76	12
13	13	157	93	29	173	109	45	189	125	61	205	141	77
14	78	14	158	94	30	174	110	46	190	126	62	206	142
15	143	79	15	159	95	31	175	111	47	191	127	63	207

by an odd number is quite difficult to implement in hardware, this has been the major argument advanced for rejecting the use of prime memory systems for vector processors.

In this note, we have pointed out that the least significant bits of the address may be used as local displacement function and allows to use the whole memory space.

Then we can state:

"Prime Memory Systems Do Not Require Euclidean Division By a Prime Number"

References

[1] P.Budnick, D.Kuck "The organization and use of parallel memories" IEEE Transaction On Computers, Dec. 1971

[2] B. Dupont de Dinechin,"A Ultra Fast Euclidean Division Algorithm for Prime Memory Systems", Supercomputing 91, Nov. 1991

[3] J.M. Frailong, W.Jalby, J.Lenfant "XOR-schemes: a flexible organization in parallel memories" Proceedings of 1985 International Conference on Parallel Processing, Aug. 1985

[4] D.J.Kuck, R.A.Stokes, "The Burroughs Scientific Processor (BSP)" IEEE Transactions on Computers, May 1982.

[5] D.T. Harper, J.R. Jump "Performance evaluation of vector accesses in parallel memories using a skewed storage scheme ", Proceedings of the 13^{th} International Symposium on Computer Architecture, June 1986

[6] D.T. Harper, J.R. Jump "Vector accesses in parallel memories using a skewed storage scheme " IEEE Transactions on Computers, Dec. 1987

[7] D.H. Lawrie "Access and alignment of data in array computer" IEEE Transactions on Computers, Dec. 1975

[8] D.H. Lawrie, C.R. Vora "The prime memory system for array access" IEEE Transactions on Computers, Dec. 1975

[9] A.Norton, E.Melton "A class of boolean linear transformations for conflict-free power-of-two stride access", Proceedings of the International Conference on Parallel Processing, 1987

[10] B.Rau, M.Schlander, D. Yen " The Cydra 5 stride insensitive memory system", Proceedings of the International Conference on Parallel Processing, 1989

[11] H.Tamura, Y.Shinkai, F.Isobe "The Supercomputer FACOM VP system" Fujitsu Sc. Tech. J., March 1985

LISTE DES DERNIERES PUBLICATIONS INTERNES IRISA

PI 635 A NOTE ON CHERNIKOVA'S ALGORITHM
Hervé LE VERGE
Février 1992, 28 pages.

PI 636 ENSEIGNER LA TYPOGRAPHIE NUMERIQUE
Jacques ANDRE, Roger D. HERSCH
Février 1992, 26 pages.

PI 637 TRADE-OFFS BETWEEN SHARED VIRTUAL MEMORY AND MESSAGE-PASSING ON AN iPSC/2 HYPERCUBE
Thierry PRIOL, Zakaria LAHJOMRI
Février 1992, 26 pages.

PI 638 RUPTURES ET CONTINUITES DANS UN CHANGEMENT DE SYSTEME TECHNIQUE
Alan MARSHALL
Mars 1992, 510 pages.

PI 639 EFFICIENT LINEAR SYSTOLIC ARRAY FOR THE KNAPSACK PROBLEM
Rumen ANDONOV, Patrice QUINTON
Mars 1992, 20 pages.

PI 640 TOWARDS THE RECONSTRUCTION OF POSET
Dieter KRATSCH, Jean-Xavier RAMPON
Mars 1992, 22 pages.

PI 641 MADMACS : A TOOL FOR THE LAYOUT OF REGULAR ARRAYS
Eric GAUTRIN, Laurent PERRAUDEAU
Mars 1992, 12 pages.

PI 642 ARCHE : UN LANGAGE PARALLELE A OBJETS FORTEMENT TYPES
Marc BENVENISTE, Valérie ISSARNY
Mars 1992, 132 pages.

PI 643 CARTESIAN AND STATISTICAL APPROACHES OF THE SATISFIABILITY PROBLEM
Israël-César LERMAN
Mars 1992, 58 pages.

PI 644 PRIME MEMORY SYSTEMS DO NOT REQUIRE EUCLIDEAN DIVISION BY A PRIME NUMBER
André SEZNEC, Yvon JEGOU, Jacques LENFANT
Mars 1992, 10 pages.

PI 645 SKEWED-ASSOCIATIVE CACHES
André SEZNEC, François BODIN
Mars 1992, 20 pages.

PI 646 INTERLEAVED PARALLEL SCHEMES : IMPROVING MEMORY THROUGHTPUT ON SUPERCOMPUTERS
André SEZNEC, Jacques LENFANT
Mars 1992, 14 pages.

PI 647 COMMUNICATING PROCESSES AND FAULT TOLERANCE : A SHARED MEMORY MULTIPROCESSOR EXPERIENCE
Michel BANATRE, Maurice JEGADO, Philippe JOUBERT, Christine MORIN
Mars 1992, 40 pages.

ISSN 0249-6399