Access to Vectors in Multi-Module Memories

Mateo Valero, Montse Peiron and Eduard Ayguadé

Departament d'Arquitectura de Computadors, Universitat Politècnica de Catalunya
Gran Capità s/num. Mòdul D6, 08071 - Barcelona (Spain).

Abstract

The poor bandwidth obtained from memory when conflicts arise in the modules or in the interconnection network degrades the performance of computers.

Address transformation schemes, such as interleaving, skewing and linear transformations, have been proposed to achieve conflict-free access for streams with constant stride. However, this is achieved only for some strides. In this paper, we summarize a mechanism to request the elements in an out-of-order way which allows to achieve conflict-free access for a larger number of strides.

We study the cases of a single vector processor and of a vector multiprocessor system. For this latter case, we propose a synchronous mode of accessing memory that can be applied in SIMD machines or in MIMD systems with decoupled access and execution.

1: Introduction

One of the factors that limits the performance of high performance computers is the latency of the memory system. In order to reduce the cost of the memory and to exploit locality in programs, the memory is organized in several levels in a hierarchical way (registers, cache memory, main memory and secondary memory).

The organization and management of the memory is strongly related with the kind of system and processor used. For instance, in a vector uniprocessor system it is not usual to have the cache memory level for vector accesses. On the other hand, in the DEC microprocessor-based workstations there are two levels of cache memory, with the lowest one integrated within the Alpha processor chip.

In this work we are concerned about the organization and access management of the main memory in vector uniprocessor and multiprocessor systems, although the techniques developed here can be used in a wider range of systems. In both cases, we assume that the cache level does not exist and we focus on the efficient access to vectors or streams. In these systems, a stream is identified by the address of its first element A_0, the length or number of elements L, and the stride or distance between each pair of consecutive elements, S.

With the goal of increasing the bandwidth between the processor and the main memory, the memory is organized in M modules (usually a power of two, 2^m). If the access latency to a memory module is $T = 2^t$ processor cycles, the memory can provide a maximum bandwidth of M/T accesses per cycle. Therefore, in a system with P processors and one port per processor a minimum of P·T modules are required to achieve one access per cycle per processor. The memory system is named matched when M = P·T and unmatched when M > P·T.

The organization of the main memory in modules requires a mechanism that assigns to each address $A = a_{n-1}a_{n-2}..a_2a_1a_0$ generated by the processor a pair <d, m>, where m is the memory module where the address is mapped and d is the displacement inside this module. The different one-to-one transformations are known as storage schemes, and they must be fast and easy to implement.

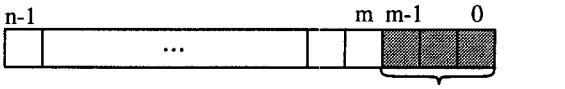
1.1: Single processor

In what follows we explain the features of the different storage schemes that have been proposed in the literature and their behaviour when accessing streams in uniprocessor systems. We assume that the processor is connected to the memory system through a single bus with a bandwidth of one request per processor cycle, so the bandwidth required from memory is one element per cycle. This bandwidth is achieved if there are no conflicts in the memory modules. A conflict occurs in the memory whenever a request is sent to a busy memory module because of a previous request, that is, whenever the separation between two consecutive requests to the same memory module is less than T cycles.

The interleaved storage scheme has been widely used because of its simplicity. In this scheme, the m least significant bits of each address indicate the memory module where the address is mapped, and the remaining bits indicate the displacement within the memory module.

0-8186-5370-1/94 $3.00

Figure 1 shows an example of a storage scheme for a system with M=8 memory modules. The calculation of the mapping does not require any additional hardware. For this reason this scheme has been used in pipelined vector computers and for the organization of main memory in less powerful computers to minimize the miss penalty and replacement in cache-based systems.



module	0	1	2	3	4	5	6	7
	0	1	2	3	4	5	6	7
	8	9	10	11	12	13	14	15
	16	17	18	19	20	21	22	23
	24	25	26	27	28	29	30	31
	⋮							

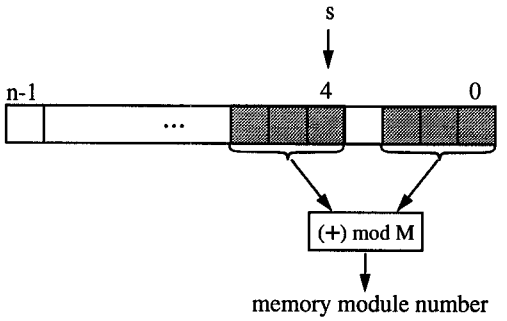
Figure 1: Interleaving storage scheme with M = 8.

The bandwidth that an interleaved memory can provide when accessing streams depends on the stride of the stream, S, on the number of memory modules, $M = 2^m$, and on the latency of the memory modules, $T = 2^t$. If the system in Figure 1 is matched (T = M = 8), only the streams with odd strides can be accessed without conflicts. This scheme classifies the strides into families such that for two streams with strides belonging to the same family, the behaviour of the memory in terms of bandwidth is the same. Each family x contains the strides $S = \sigma \cdot 2^x$, where σ is an odd number [7].

Increasing the number of memory modules increases the number of families of strides that can be accessed without conflicts, so that in general there are m-t+1 families of conflict-free strides (x = 0, 1, .., m-t). On the other hand, the use of buffers in the memory modules or in the processor does not minimize the access time for a stream with a non conflict-free stride.

Another kind of storage schemes are skewing or block-row rotations [2]. They were initially proposed and used for array processors, but have also been proposed for pipelined vector processors. An example for M = 8 is shown in Figure 2.

Given an address A, the module where it is mapped is obtained by adding modulo M two fields of m bits. The location of the upper field defines the behaviour of the storage scheme. The displacement inside the module is indicated by the most significant n-m bits of the address.



module	0	1	2	3	4	5	6	7
	0	1	2	3	4	5	6	7
	8	9	10	11	12	13	14	15
	23	16	17	18	19	20	21	22
	31	24	25	26	27	28	29	30
	38	39	32	33	34	35	36	37
	⋮							

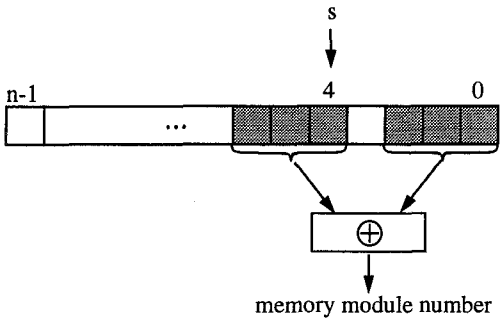
Figure 2: Skewing storage scheme with M = 8 and s = 4.

When accessing streams, skewing-based storage schemes allow conflict-free access to the same number of families than with interleaved schemes, but now the families can be different. For instance, observe in Figure 2 that any stream with a stride of the family x = s = 4 is accessed in a conflict-free way. In general, if the lowest bit of the second field is located in bit s, the families x = s, s+1, .., s+m-t can be accessed in a conflict-free manner. Streams with strides of the families x > s have their elements located in a number of memory modules that is not sufficient to avoid conflicts, so the use of buffers gives no additional benefits. However, this is not the case when the stride belongs to families with x < s. In this case, the access latency depends on the values of A_0, L and S, and it can be reduced by the use of buffers [7].

In the storage schemes based on linear transformations, or XOR-schemes, the m bits that indicate the memory module ($b_{m-1}..b_0$) are obtained by applying an exclusive-OR function to some bits of the address. As a consequence, there exist a lot of different linear transformations. Figure 3 shows an example with M = 8 and with the linear transformation $b_i = a_i \oplus a_{i+s}$ ($0 \le i < t$ and s = 4). Observe that any stream with a stride of the family x = s = 4 is conflict-free.

The XOR-schemes have been proposed with different goals for array processors [5], multiprocessors [8], vector processors [6] and VLIW [12], and have been used in machines such RP3 [11] and Cydra [12].

The number of families that produce conflict-free strides is the same than the obtained with skewed schemes. However, the hardware that implements the XOR transformation is cheaper and faster than the one needed in skewing-based transformations.



module	0	1	2	3	4	5	6	7
	0	1	2	3	4	5	6	7
	8	9	10	11	12	13	14	15
	17	16	19	18	21	20	23	22
	25	24	27	26	29	28	31	30
	34	35	32	33	38	39	36	37
	42	43	40	41	46	47	44	45
	51	50	49	48	55	54	53	52
	59	58	57	56	63	62	61	60
	68	69	70	71	64	65	66	67

Figure 3: XOR scheme with M = 8 and s = 4.

In summary, the three types of storage schemes allow conflict-free access to the same number of families, m-t+1. The difference between them is the hardware required as well as the behaviour when accessing streams with strides belonging to families that are not conflict-free.

1.2: Vector multiprocessors

When several processors or ports are considered, the organization of the memory system is significantly more complicated than the one for a single processor with a single port. Typically, the memory is composed of $M = 2^m$ memory modules grouped into 2^s sections; the 2^{m-s} modules in each section are connected by a single bus and the processor ports are connected to the sections through an interconnection network, as shown in Figure 4. This organization allows the initiation each processor cycle of one access per section, as long as the requested module is not busy with a previous request.

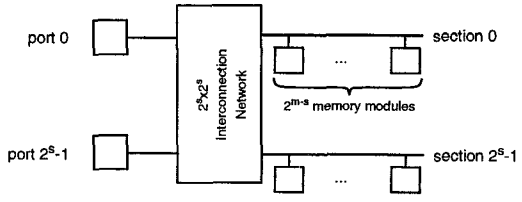
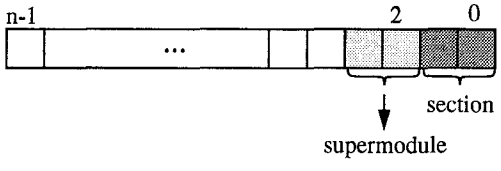


Figure 4: Structure of the system.

With this organization, the address mapping obtains, from the address A, the tuple (section, supermodule) instead of memory module number. The term supermodule refers to the module number within a section. Interleaving has been widely used in current systems. Figure 5 shows an interleaved storage scheme for a memory system with M=16 memory modules and 4 sections (s=2). In this scheme, the lowest 2 bits of the address are used to map the address into a section and the next 2 bits to map it into a supermodule.

Since each port performs asynchronous access to memory, it is seldom possible to obtain a conflict-free access to the memory system. Even when the distribution of the elements of individual streams allow a conflict-free access, the collisions in the network and the conflicts in the memory modules, caused by the other requests, introduce additional latencies in the access and, therefore, a loss of efficiency in the overall system. The corresponding evaluation is very complicated and has been approached by several methods [1, 3, 4, 9, 14].

Recent researchers have provided attention to the synchronous access to several vectors. For instance, in [13] vectors are assumed to belong to a single stream and the processors perform accesses in a synchronized manner. These assumptions are reasonable in SIMD vector multiprocessors or in MIMD systems with decoupled access, in which the data can be accessed in this regular and



section	0				1				2				3			
supermodule	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
	0	4	8	12	1	5	9	13	2	6	10	14	3	7	11	15
	16	20	24	28	17	21	25	29	18	22	26	30	19	23	27	31
	⋮	⋮						⋮								
	112	116	120	124	113	117	121	125	114	118	122	126	115	119	123	127
								⋮								

Figure 5: Interleaving storage scheme for a system with m=4 and s=2.

synchronized manner, but then used differently. In this paper we also describe an alternative solution to this problem.

Up to now we have assumed that the stream elements are requested in order. In this paper we describe an out-of-order mechanism to request the elements of the stream that increases the number of families of strides for which conflict-free access is obtained. In Section 2 we describe the conditions that have to be fulfilled by the streams to obtain conflict-free access when the elements are requested out-of-order. Section 3 describes the application of this technique to the case of a uniprocessor system with both matched and unmatched memory systems. We also describe the hardware needed to implement the out-of-order mechanism and show that it is of similar complexity to the one required with ordered access.

Section 4 of this paper is devoted to the case of a multiprocessor system with several requesting ports. A storage scheme and a synchronized access method to vectors belonging to a single stream is described. Finally, we conclude the paper and present some future work.

2: Balanced streams

Assume the architecture model shown in Figure 6 of a uniprocessor system connected to M memory modules with latency $T = 2^t$ through a single bus. The processor requests the elements of a stream defined by the address of the first element, A_0, the length of the stream or number of elements, $L = 2^\lambda = k \cdot T$, and the stride $S = \sigma \cdot 2^x$.

Assuming that the processor sends a new address each cycle, conflict-free access means that the memory finishes the service to the L elements in L+T-1 cycles. To achieve this it is sufficient that two consecutive requests to any memory module are separated by at least T cycles; we say that a sequence of addresses is conflict free if it satisfies this condition. If the condition is not satisfied, a request is sent to a busy memory module, and the corresponding delay is added to the memory access latency.

A stream is balanced if each memory module has a number of elements less or equal than L/T. So, if the memory system is matched a stream is balanced if each memory module contains exactly L/M stream elements.

For a conflict-free access it is necessary that the stream is balanced. This is a necessary but not sufficient condition. If a stream is balanced it is always possible to obtain an access ordering that leads to a minimum access time for the whole stream.

Since σ is odd, the addresses of the $L = 2^\lambda$ stream elements have different values in the λ bits $x+\lambda-1..x$. The bits $x-1..0$ have the same value for all the elements, and the values of the bits $n-1..x+\lambda$ depend on A_0 and S. The stream is balanced if the bits between $x+\lambda-1..x$ decide at least t bits of the memory module number.

For instance, in the scheme in Figure 1 just the streams with strides of the family x = 0 (odd strides) are balanced. For $x > 0$, the elements are located in $\lceil M/2^x \rceil$ memory modules and as a consequence the stream is not balanced.

For the skewing and XOR-transformation storage schemes shown in Figures 2 and 3, the maximum number of balanced families is $s+1$ if $\lambda > s+t-1$ and $\lambda-t+1$ if $\lambda \leq s+t-1$. So, at most $\lambda-t+1$ families of strides are balanced with these two schemes. The same situation occurs with the block-interleaved storage scheme, in which the bits indicating the module number are any t consecutive bits $s+t-1..s$ instead of the t least significant bits.

In the next section we describe an access ordering method that allows conflict-free access to streams with a stride belonging to the balanced families of strides.

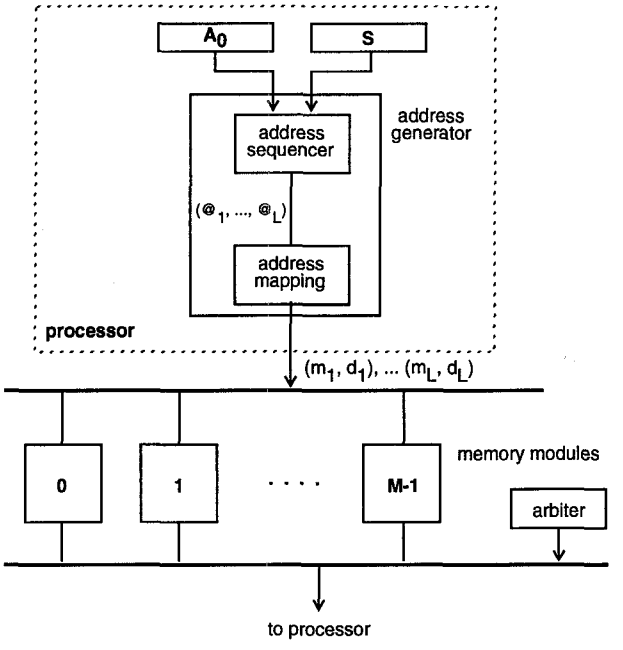


Figure 6: Architecture model.

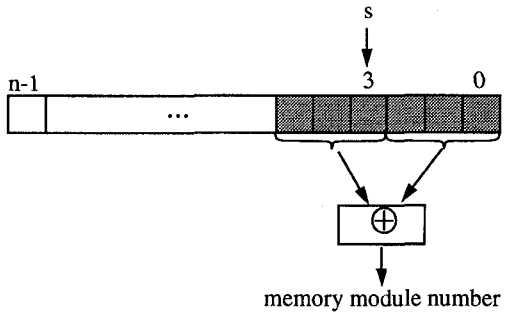
3: Single processor

3.1: Matched-memory case

Assume the XOR storage scheme shown in Figure 7, with $M = T = 8$ and the access to a stream with $L = 2^\lambda = 64$. From the previous section we conclude that the families x = 0, 1, 2 and 3 result in balanced streams. On the one hand we know that the family x = 3 allows the stream to be accessed in order without conflicts. For instance, if A_0 is 32 and the stride is $S = 40 = 5 \cdot 2^3$, the first 8 elements are located in the

memory modules (4,1,6,3,0,5,2,7). And this succession of modules is repeated for every group of 8 elements.

On the other hand, families x = 2, 1 and 0 do not allow an access in order without conflicts. For instance, with the same value of A_0 and S = 20 = $5 \cdot 2^2$, the first 16 elements are located in the memory modules (4,7,1,4,6,1,3,6,0,3,5, 0,2,5,7,2), and the first conflict occurs between the 1st and 4th elements, both located in memory module 4. However, observe that the period for the even elements is (4,1,6,3,0,5,2,7) and the period for the odd elements is (7,4,1,6,3,0,5,2), and that each one is conflict free if considered alone. To obtain that, we have requested the elements of the stream with a stride which is conflict free, and made a set of sequences to request all the elements of the stream.



module	0	1	2	3	4	5	6	7
	0	1	2	3	4	5	6	7
	9	8	11	10	13	12	15	14
	18	19	16	17	22	23	20	21
	27	26	25	24	31	30	29	28
	36	37	38	39	32	33	34	35
	45	44	47	46	41	40	43	42
	54	55	52	53	50	51	48	49
	63	62	61	60	59	58	57	56
	64	65	66	67	68	69	70	71
	⋮							

Figure 7: XOR scheme with M = 8 and s = 3.

In general (see [15] for a detailed discussion), for a stream with $L=2^{\lambda}$ elements, any starting address and stride $\sigma \cdot 2^x$ (x<s), it is possible to construct L/T sequences of T elements each so that all the elements in each sequence are mapped into different memory modules. If the initial element of a sequence is denoted with i, the elements of the sequence are $i + k \cdot 2^{s-x}$ with k=1, 2, .., T-1. Different sequences are identified by $i = j_1 \cdot 2^{s+t} + j_0$ with $j_0 = 0, 1,.., 2^{s-x}-1$ and $j_1 = 0, 1,.., 2^{\lambda-(s+t-x)}-1$.

For the previous example, if the starting address of the stream is 32 and the stride is S = 20 = $5 \cdot 2^2$, the memory modules referenced by the resulting sequences are (4,1,6,3,0,5,2,7), (7,4,1,6,3,0,5,2) and this is repeated three more times (if they are requested in lexicographical order). Observe that the first element of sequence 1 (accessing memory module 7) collides with the last element of sequence 0. Therefore, with the previous out of order access we obtain sequences that access memory modules in a conflict-free way, but there might be conflicts between two successive sequences.

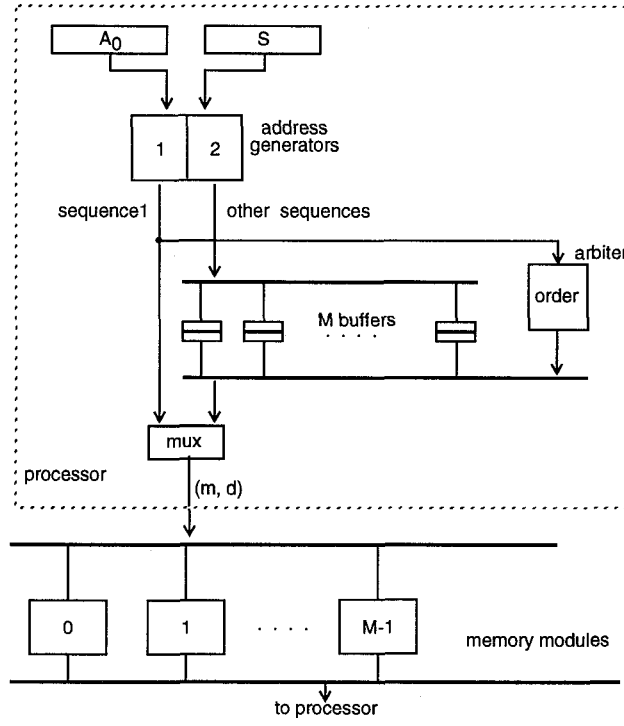


Figure 8: Architecture model for out-of-order access

To avoid intersequence conflicts, it is necessary to incorporate a second reordering so that the order in which modules are visited by all sequences is the same. The module order of the first sequence is used as reference to access the elements in the other sequences. For the previous example, the module order of the first sequence is (4,1,6,3,0,5,2,7) so that the order of access of the second sequence has to be (4,1,6,3,0,5,2,7). However, this poses a problem with the calculation of the addresses inside the sequences, since to have a simple incremental calculation (adding $\sigma \cdot 2^s$) it is necessary to do this in the order described in the previous section. The solution is to decouple the calculation of the addresses from the actual requests. This is achieved by calculating the addresses of sequence i+1 while accessing sequence i. In this way, when the access to sequence i+1 is started, the addresses of all its elements have been calculated and can be sent to memory in the appropriate order. If no cycles have to be lost, during the first 2^t cycles, it is necessary to calculate the addresses of the first sequence (which are used immediately for memory

access) and of the second sequence (which are stored in a set of latches for access as the next sequence). Moreover, it is necessary to store the module order of the first sequence, which is used to control the order of the requests of the following sequences. After that, for each sequence, the addresses for access are obtained from the latches and a new addresses are calculated to store. Consequently, as shown in Figure 8, two address generators are needed, although one of them is only used in the first 2^t cycles. In addition to the latches in the processor, neither buffers are needed in the memory modules nor an arbiter in the return bus to the processor.

3.2: Unmatched-memory case

One way to increase the number of families of strides that produce conflict-free access, is to increase the number of memory modules (m > t). The particular case of m=2·t is presented in this section. For the general case of m=k·t+k' the reader is referred to [15].

The address mapping we use is the following

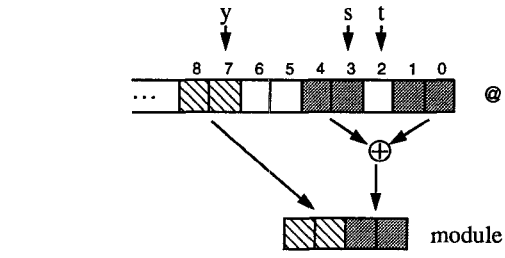
$$b_i = \begin{cases} a_{s+i} \oplus a_i & 0 \le i \le t-1,\ s \ge t \\ a_{y+i-t} & t \le i \le 2t-1,\ y \ge s+t \end{cases}$$

where a XOR linear transformation is applied to two fields of t bits in the address and the rest of bits to compute the memory module are taken from a field of m-t bits in the address. An example for t = 2, m = 4, s = 3 and y = 7 is given in Figure 9.

This address mapping corresponds to a division of the memory modules into T sections of T modules each and of the address space into blocks of 2^y locations; each block is mapped into one section, using the mapping defined by the lower t bits of b.

With this mapping, and using a similar out-of-order access method as in the matched case, the maximum number of conflict-free families is obtained when s = λ-t and y=2·(λ-t)+1. This results in conflict-free access for $0 \le x \le 2\cdot(\lambda-t)+1$. If the access is performed in order, just families x=y and x=s are conflict-free. To achieve conflict-free access for the balanced families, we divide the stream into sequences and perform the access in an out-of-order way. Now depending on the value of x (x≤s or s<x≤y), the elements within a sequence are requested with a stride $\sigma\cdot 2^s$ or $\sigma\cdot 2^y$, respectively, so all the elements in a sequence are mapped into a different memory module.

To avoid intersequence conflicts, a slightly modified strategy than the one proposed in Section 3.1 for the matched memory case is used [15].



module	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	0	1	2	3	128	129	130	131	256	257	258	259	384			
	4	5	6	7	132	133	134	135	260	261	263	262				
	9	8	11	10	137	136	139	138								
	13	12	15	14	141	140	143	142								
	18	19	16	17	146											
	22	23	20	21												
	27	26	25	24	155											
	31	30	29	28												
	32				160				288				416			
	41															
	50															
	59															
	64				192				320				448			
	73															
	84															
	93															
	96				224				352				480			
	105															
	114															
	123	122	121	120									507	506	505	504
	127	126	125	124									511	510	509	508
	512	513	514	515	640				768				896			
	516	517	518	519												

Figure 9: XOR-based transformation and mapping when m=4, t=2, s=3 and y=7.

For instance, in a system with M=16 and T=4 (e.g. Cray-1) we obtain conflict-free access to families x=0..9. If an interleaved storage scheme and access in order are used it is possible to access families x=0..2 without conflicts.

4: Vector multiprocessor

In this section we study the behaviour of memory accesses in a multiprocessor system with the structure shown in Figure 4. We assume that the memory system is matched, that is, that the latency of the memory modules is $T=2^t$ and the number of memory modules is $2^m=2^{s+t}$. The maximum achievable throughput is then $P=2^s$ accesses per processor cycle after the initial transient state.

With the sake of clarity, we assume that the interconnection network is a crossbar, although the method applies also for the case of having an Omega interconnection network (see [10]).

The ports are controlled by vector load/store instructions and interface with the processors through vector registers of length $L = 2^{\lambda}$. As shown in Figure 10, port i accesses vector V_i composed of L consecutive elements of the stream. The stride of the stream is S and the initial address is A_0.

We now determine necessary conditions to allow conflict-free access for the whole stream. In the multiprocessor architecture model described in Section 1.2, two types of conflicts prevent conflict-free access: memory module conflicts (which occur when a request arrives to a module while it is busy) and section conflicts (which occur when two simultaneous requests are for the same section).

As in the single processor case, a necessary condition to avoid memory module conflicts is that each memory module does not contain more than L/T elements of the stream, because a conflict-free access has to last for L+T-1 memory cycles, and this is not possible if a module has more than L/T elements. Similarly, to avoid section conflicts, each section has to contain the same number of stream elements. When these two conditions are satisfied, we say that stream is balanced.

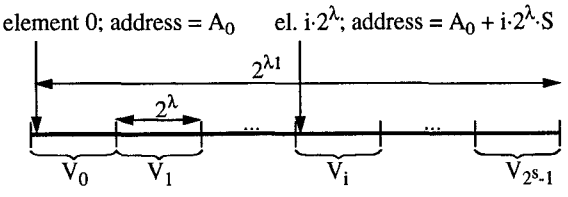


Figure 10: Vectors in a stream.

We use the storage scheme shown in Figure 11 (known as block-interleaved), where the S-field of s bits specifies the section, the M-field of t bits specifies the supermodule and the rest of the bits of the address specify the displacement inside the module.

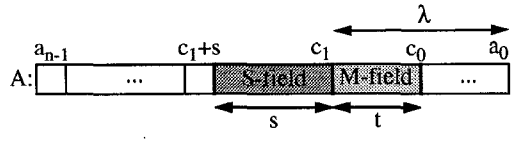


Figure 11: Storage scheme used in this paper.

This location of the M-field and the S-field produces the largest set of balanced families including odd strides: x=0, 1, .., c_0 (see [10] for the proof).

Once a set of balanced families is obtained, a conflict-free access ordering has to be proposed. Two conditions have to be satisfied:

C1: The P simultaneous requests must not have section conflicts.

C2: Consecutive accesses to a memory module have to be separated by T cycles.

One stream may have many conflict-free access orderings; the access ordering that we propose satisfies the following properties:

P1: Simultaneous accesses go to different sections (this is necessary, and equivalent to C1).

P2: Simultaneous accesses go to the same supermodule.

P3: T consecutive accesses go to different supermodules.

P2 and P3 are a particular way of satisfying condition C2 and result in a simple hardware for address calculation.

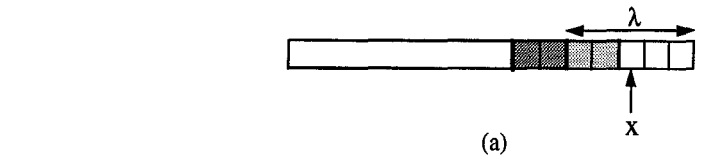
We now determine an out-of-order accessing scheme that satisfies these conditions for the balanced families of the address mapping of Figure 11. To achieve this we divide each vector into L/T sequences of T elements each, in a similar way we proposed for the single processor case. The elements of the stream are identified by the triple: vector number i ($0 \le i \le P-1$), sequence number j ($0 \le j \le L/T-1$), and element number k ($0 \le k \le T-1$).

Consider the address mapping shown in Figure 12.a. Figure 12.b shows one period of the address mapping and Figure 12.c shows how the addresses of the elements of a stream with A_0=4 and S=4 (σ =1 and x=2) are mapped and their grouping into vectors. Observe that, for instance, the elements k and (k+1) with k odd of each vector are in the same memory module, so they cannot be requested in successive cycles; moreover, the simultaneous requests are to the same section, so conflicts arise in the interconnection network too.

Now suppose that the elements are accessed by sequences as shown in Figure 12.d; it can be seen that properties P1, P2 and P3 are fulfilled for the simultaneously requested sequences, so the access is conflict-free.

Although each sequence is accessed without conflicts, the access of consecutive sequences might lead to conflicts in the memory modules. In Figure 12.d, for instance, supermodules are visited in the order <0,1,2,3> by the first sequences and in the order <1,2,3,0> by the second ones; all the requests made in the fifth cycle collide with those made in the second cycle.

To solve these intersequence conflicts, we use two address generators, as in the single processor case. During the first T cycles, the addresses of the first sequence are calculated and used for memory access; in addition, its supermodule order is stored in a circular shift register. Meanwhile, the addresses of the second sequence are calculated and stored in a set of buffers. After that, for each



(a)

section	0				1				2				3			
supermodule	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
one period	0	8	16	24	32	40	48	56	64	72	80	88	96	104	112	120
	1	9					⋮									
	⋮	⋮														
	7	15	23	31	39	47	55	63	71	79	87	95	103	111	119	127
	128	136	144	152	160	168	176	284	192	200	208	216	224	232	240	248
							⋮									

(b)

section	0				1				2				3			
supermodule	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
V_0		8	16	24	32	40	48	56	64	72	80	88	96	104	112	120
	4	12	20	28	36	44	52	60	68	76	84	92	100	108	116	124
	128	136	144	152	160	168	176	184	192	200	208	216	224	232	240	248
V_1	132	140	148	156	164	172	180	188	196	204	212	220	228	236	244	252
	256	264	272	280	288	296	304	312	320	328	336	344	352	360	368	376
V_2	260	268	276	284	292	300	308	316	324	332	340	348	356	364	372	380
	384	392	400	408	416	424	432	440	448	456	464	472	480	488	496	504
V_3	388	396	404	412	420	428	436	444	452	460	468	476	484	492	500	508
	512															

(c)

section	0				1				2				3			
supermodule	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
V_0		8	16	24	32	40	48	56	64	72	80	88	96	104	112	120
	4	12	20	28	36	44	52	60	68	76	84	92	100	108	116	124
	128	136	144	152	160	168	176	184	192	200	208	216	224	232	240	248
V_1	132	140	148	156	164	172	180	188	196	204	212	220	228	236	244	252
	256	264	272	280	288	296	304	312	320	328	336	344	352	360	368	376
V_2	260	268	276	284	292	300	308	316	324	332	340	348	356	364	372	380
	384	392	400	408	416	424	432	440	448	456	464	472	480	488	496	504
V_3	388	396	404	412	420	428	436	444	452	460	468	476	484	492	500	508
	512															

: first sequences : second sequences : third sequences

(d)

Figure 12: (a) Storage scheme, (b) Address mapping, (c) Mapping of the elements of a stream with $A_0 = 4$ and $S = 4$, and (d) Sequences in the stream.

sequence, the addresses for access are obtained from the buffers under the control of the shift register, and new addresses are calculated to store (so, the second address generator works only during the first T cycles).

5: Conclusions

In this paper we have summarized an out-of-order access mechanism to obtain conflict-free access to more families of strides than the access in order. We have considered the access of streams of fixed length, equal to the length of a vector register. The access patterns correspond to constant strides and the stream can begin in any address. The basic idea we propose is an out-of-order access of the elements of the stream to achieve conflict-free access for all strides that produce balanced streams. We consider matched and unmatched memory systems as well as single and multiple vector processors.

In the single processor case, we obtain a window of λ-t+1 families of strides that are conflict free if the memory is matched, whereas previous schemes that perform the access in order result in a single conflict-free family (for vectors of any length). For an unmatched memory systems with $M=T^2$, the size of the conflict-free window is doubled; this compares favourably to the t+1 conflict-free families obtained with ordered access. The method can be applied to memory systems with any degree of unmatchness ($m=k \cdot t+k'$) obtaining conflict-free access to $k \cdot (\lambda-t+1)+k'$ families. As shown in [15], more conflict-free families can be obtained at a cost of complicating somewhat the sequence generation for the additional families.

In the case of a vector multiprocessor, the number of conflict-free families is λ-t+1 if the memory system is matched and the different processors synchronously access vectors of a common stream. With an unmatched memory system [10] it is possible to add one conflict-free family each time we double the number of modules, up to t additional conflict-free families.

Acknowledgments

This work has been supported by the ESPRIT III Basic Research Action 6634 (APPARC), the Ministry of Education of Spain under contract TIC-880/92, and by the CEPBA (European Centre for Parallelism of Barcelona).

References

1. D.H. Bailey, "Vector Computer Memory Bank Contention", IEEE Trans. on Computers, vol. 36, no. 3, pp. 293-298, 1987.
2. P. Budnik and D. J. Kuck, "The Organization and Use of Parallel Memories", IEEE Trans. on Computers, vol. 20, no. 12, pp. 1566-1569, 1971.
3. D.A. Calahan, "Some Results in Memory Conflict Analysis", Proc. of Supercomputing'89, pp. 775-778, 1989.
4. T. Cheung and J.E. Smith, "A Simulation Study of the Cray X-MP Memory System", IEEE Trans. on Computers, vol. 35, no. 7, pp. 613-622, 1986.
5. J. Frailong, W. Jalby and J. Lenfant, "XOR-schemes: A Flexible Data Organization in Parallel Memories", Int'l Conference on Parallel Processing, pp. 276-283, 1985.
6. D.T. Harper III, "Block, Multistride Vector and FFT Accesses in Parallel Memory Systems", IEEE Trans. on Parallel and Distributed Systems, vol. 2, no. 1, pp. 43-51, 1991.
7. D.T. Harper III and D. A. Linebarger, "Conflict-Free Vector Access Using a Dynamic Storage Scheme", IEEE Trans. on Computers, vol. 40, no. 3, pp. 276-283, 1991.
8. A. Norton and E. Melton, "A Class of Boolean Linear Transformations for Conflict-Free Power-of-Two Stride Access", Int'l Conference on Parallel Processing, pp. 247-254, 1987.
9. W. Oed and O. Lange, "On the Effective Bandwidth of Interleaved Memories in Vector Processing Systems, IEEE Trans. on Computers, vol. 34, no. 10, pp. 949-957, 1985.
10. M. Peiron, M. Valero, E. Ayguadé and T. Lang, "Conflict-Free Access to Streams in Multiprocessor Systems", DAC/ UPC Research Report 93/04, 1993.
11. G.F. Pfister et al., "The IBM Research Parallel Processor Prototype (RP3): Introduction and Architecture", Int'l Conference on Parallel Processing, pp. 764-771, 1985.
12. B. R. Rau, M. S. Schlansker and D. W. L. Yen, "The Cydra™ 5 Stride-Insensitive Memory System", Int'l Conference on Parallel Processing, pp. 242-246, 1989.
13. A. Seznec and J. Lenfant, "Interleaved Parallel Schemes: Improving Memory Throughput on Supercomputers", Int. Symp. on Computer Architecture, pp. 246-255, 1992.
14. J.E. Smith and W.R. Taylor, "Characterizing Memory Performance in Vector Multiprocessors", Int. Conf. on Supercomputing, pp. 35-44, 1992.
15. M. Valero, T. Lang, M. Peiron and E. Ayguade, "Conflict-Free Access for Streams in Multi-module Memories", IEEE Trans. on Computers, (to appear), 1994.