Conflict free accesses to strided vectors on a banked cache
appears in IEEE Transactions on Computers, July 2005

André Seznec
IRISA/INRIA
Rennes France

Roger Espasa
UPC/Intel Labs
Barcelona Spain

ABSTRACT

With the advance of integration technology, it has become feasible to implement a microprocessor, a vector unit and a multimegabyte bank-interleaved L2 cache on a single die.

Parallel access to strided vectors on the L2 cache is a major performance issue on such vector microprocessors. A major difficulty for such a parallel access is that one would like to interleave the cache on a block size basis in order to benefit from spatial locality and to maintain a low tag volume, while strided vector accesses naturally work on a word granularity.

In this paper, we address this issue. Considering a parallel vector unit with 2^n independent lanes, a 2^n bank interleaved cache and a cache line size of 2^k words, we show that, any slice of 2^{n+k} consecutive elements of any strided vector with stride $2^r R$ with R odd and $r \leq k$ can be accessed in the L2 cache and routed back to the lanes in 2^k subslices of 2^n elements.

KEYWORDS

Vector microprocessor, strided vectors, conflict free access, L2 caches

I. INTRODUCTION

Nowadays, the integration technology is exploiting the extensive use of cache hierarchies, even on processors targeted to scientific computing. Pipelined access to a bank-interleaved main memory to feed a vector execution core is no longer a cost-effective solution. It has now become feasible to implement a single chip a powerful superscalar microprocessor coupled with a vector unit and a large multimegabyte on-chip cache as illustrated by the Tarantula project [2].

Parallel access to strided vectors in the L2 cache is a major performance issue on such a vector microprocessor. Parallel access to vectors in main memory was also a major issue for vector supercomputers. Therefore, parallel access to strided vectors in bank interleaved memories has been abundantly studied in the literature till the mid 90's [1], [4], [3], [6], [5]. However, these solutions only work for memories (or caches) interleaved on a per word basis and cannot be directly extended to memories (or caches) interleaved on a wider granularity. On a vector microprocessor, one would like to interleave the cache on a block size basis (i.e., 64 bytes on Tarantula) in order to benefit from spatial locality and to maintain a low tag volume.

On the Tarantula processor, conflict free access on the L2 cache for any strided vector with stride of the form $2^k\ R$ with R odd and $k \leq 3$ was possible. In this article, we present the underlying theory that was used in the design of Tarantula [2] for enabling full cache bandwidth use on vector accesses. More precisely, we show that, considering a parallel vector unit with 2^n independent lanes, a 2^n bank interleaved cache and a cache line size of 2^k words, any slice of 2^{n+k} consecutive elements of any strided vector with stride $2^r R$ with R odd and $r \leq k$ can be accessed in the L2 cache and routed to the lanes in 2^k subslices of 2^n elements.

The remainder of this article is organized as follows. In Section II we describe the architecture model we consider for the vector microprocessor. Then Section III describes the specificity of the problem of parallel access to strided vectors on a bank-interleaved cache. In Section IV, we develop the theory that enables the use of full cache bandwidth on most vector accesses. Section V discusses the practical use of this theory in the design

This work was done while the authors were with Compaq Alpha Development Group from 02-1999 to 02-2000

of Tarantula. Section VI summarizes this study.

II. Vector Microprocessor Architecture Model

In this section, we describe the vector architecture model considered in the paper.

We consider a vector microprocessor build around a superscalar execution core and a parallel vector unit consisting in N=2^n execution units or *lanes*.

The vector unit manages the accesses to vector data in memory. These accesses to memory are executed in parallel through an interleaved L2 cache as in the Tarantula project [2]. Each lane can compute a memory address and send/receive a data word per cycle. Each L2 cache bank support one access per cycle.

For the sake of simplicity, in this article, we will assume that the number of banks in the L2 cache and the number of lanes in the vector unit are equal to 2^n.

A vector register file is implemented and each lane has read and write access to only a section of the register file: if 2^{p+n} is the total number of elements in a vector register, lane x works on elements number x, $x + 2^n$, .., $x + (2^p - 1) * (2^n)$ in the vector. Therefore a vector instruction activates 2^p consecutive operations on each lane.

Note that, 2^k being the cache line size measured in words, the conflict free access property demonstrated in this article only stands when p is larger or equal than k. For the sake of simplicity, we will assume p=k through the remainder of the paper.

We summarize below the characteristics of the architecture model:

1) The number of lanes and the number of cache banks are equal to $N = 2^n$.
2) Each lane can compute one address word and may receive/send one word of data from/to the L2 cache per cycle.
3) The cache line size is CL=2^k
4) The total size of a vector register is $M = 2^{n+k}$ words, i.e. 2^k register words per lane.
5) Element V(i) of a vector register is stored in lane $i \ mod \ 2^n$, and its local index in the vector register is $\frac{i}{2^n}$.

The parallel and conflict free access properties presented in the remainder of the article can be easily extended, when the number of cache banks is a power of two multiple of the number of lanes and/or when the number of elements handled by a lane in a vector register is multiple of the size in words of the cache line.

III. Issues on parallel access to strided vectors on a bank-interleaved cache

A. The fundamental vector distribution theorem on interleaved memory

Parallel access to strided vectors have been extensively studied, but the most fundamental property on the distribution of the elements among the memory banks is known since 1971 [1].

Theorem 1 (Distribution Theorem): If a memory is interleaved on a per word basis on N banks then for any vector V stored with a stride R, $V(i)$ and $V(j)$ are stored in the same memory bank iff $i = j \bmod N/GCD(N, R)$

In particular, if N is a power of two then the access to any slice of N consecutive elements of a vector V stored with an odd stride is conflict-free.

B. Multiword block size and conflicts

The main difference between the use of an interleaved cache and the use of an interleaved memory on a supercomputer is the width of the banks. Natural interleaving on a cache is on a per cache block basis while interleaving the main memory on a vector supercomputer is on a per word basis. Using single word cache line size on a vector microprocessor is not a cost-effective solution, since a significant portion of the cache space is used for storing cache tags. As an example, when considering physical addresses on 52 bits and 8-way set-associative 8 Mbytes cache, one would use 32-bit tags for each cache word, i.e., a third of the total storage space (data and tags).

Unfortunately, when using a multiword block size, slices of 2^n consecutive elements of most strided vectors can not be accessed in parallel on the 2^n banks as illustrated on Figure 1. Therefore, using alternative solutions must be considered for accessing strided vectors on an interleaved cache.

C. Leveraging the vector granularity

On the vector microprocessor we are considering in this paper, the granularity of a vector instruction is two-fold: lane parallelism and local vector register parallelism. The granularity of a vector instruction, and in particular of a vector access on memory, is $2^n * 2^k$ elements: a vector access will activate the cache for at least 2^k cycles.

Exploiting the full bandwidth of the L2 cache on a vector access does not require it to access 2^n consecutive elements of a vector in parallel on a single cycle, but rather to access 2^{n+k} consecutive elements of a vector in 2^k consecutive cycles.

Bank 0				Bank 1				Bank 2				Bank 3			
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
32	33	34	35	**36**	37	38	39	40	41	42	43	44	**45**	46	47
48	49	50	51	52	53	**54**	55	56	57	58	59	60	61	62	**63**
64	65	66	67	68	69	70	71	**72**	73	74	75	76	77	78	79
80	**81**	82	83	84	85	86	87	88	89	**90**	91	92	93	94	95
96	97	98	**99**	100	101	102	103	104	105	106	107	**108**	109	110	111
112	113	114	115	116	**117**	118	119	120	121	122	123	124	125	**126**	127
128	129	130	131	132	133	134	**135**	136	137	138	139	140	141	142	143

Fig. 1. Sixteen consecutive elements of a vector with stride 9 and base address 0: 4 elements per cache bank

On the example illustrated on Figure 1, the sixteen elements of the vector V stored with base address 0 and stride 9 are equitably distributed among the four cache banks. However the vector elements cannot be accessed in parallel in the natural index order since elements V(0) and V(2) are stored in the same cache bank at respective address 0 et 18. Organizing the subslices to access one element per cache bank is also not sufficient. For instance, V(0), V(4), V(8), V(12) lie in the four distinct cache banks, but must be processed by the same lane: conflicts will occur when routing the words to the lanes. However, the accesses can be organized in such a way that, on each cycle one word is accessed on each bank and that one of these words will be processed on each lane: for example, V(0), V(1), V(6), V(7) on the first slice (respective banks 0,2,1,3) V(4), V(5), V(2), V(3) on the second slice, V(8), V(9), V(14), V(15) on the third slice and V(12), V(13), V(9), V(10) on the fourth slice.

More formally, for accessing a slice of $2^n * 2^k$ consecutive elements of the vector in the cache in the minimum possible delay, the cache will be able to deliver a throughput of 2^n words per cycle if the accesses to the different elements of the vector can be organized in such a way that the limits of two material constraints are reached:

1) Each bank receives one request per cycle.
2) Each lane sends one request per cycle.

Note that using a vector granularity larger than the number of lanes (or processors) to enable conflict-free access to large families of vectors on word-interleaved memories was also considered in [6], [5].

In the next section, we construct such a partition of the access to the cache banks for a large family of strided vectors.

D. Execution of subsequent vector instructions

Our example illustrates that vector elements return to their respective lanes out-of-order. Apart loads and stores, vector instructions process their data in the natural order of the indices. Therefore, in order to execute subsequent dependent vector instructions after a load, the vector unit must wait until all the vector elements have been routed back to the lanes and stored in the target vector register.

IV. Accessing strided vectors on a multiword bank interleaved cache

In this section, we show that, provided that vector elements are accessed out of (the natural) order, the full bandwidth of the cache can be exploited for the accesses to a large family of vectors.

Theorem 2 (Fundamental Theorem): Let $N = 2^n$ be the number of lanes and of cache banks,
let CL=2^k be the cache line size,
let 2^k be the number of elements handled by each lane in a vector register
let V be a vector,with a stride of the form $2^r R$ with R odd and $r \leq k$,
then for any arbitrary slice of 2^{n+k} consecutive elements of vector V, there exists a partition of the slice in 2^k subslices S_i of 2^n elements, each subslice S_i being both lane conflict free (i.e. a single request is issued by each lane) and cache bank conflict free (i.e., each bank receives only one request).

The proof of the theorem is constructive and allows to compute such a partition for any vector V respecting Theorem 2 hypothesis. As shown in Section V, it allows the realistic implementation of a vector access unit.

In order to prove the theorem, we first prove the following lemma.

A. Fundamental lemma

Let us call the E^p the set $E^p = \{0, .., 2^p - 1\}$

Lemma 1 (Fundamental Lemma): Let b,k,n be positive integers, let R be an odd integer, then there exists a function $f_{n,k}^{R,b}$ from $E^k * E^n$ to E^k such that

1) for any X in E^k
the function $g_{n,k}^{R,b,X}$ defined below is a bijection:
$$g_{n,k}^{R,b,X} : \begin{array}{ccc} E^n & \rightarrow & E^n \\ i & \rightarrow & \frac{b+R*(i+2^n*f_{n,k}^{R,b}(X,i))}{2^k} \ mod\ 2^n \end{array}$$
2) for any I in E^n, the function $h_{n,k}^{R,b,I}$ defined by $h_{n,k}^{R,b,I}(X) = f_{n,k}^{R,b}(X,I)$ is a bijection from E^k on itself.

Some intuitive explanation of Lemma 1 use: Let V be a vector with an odd stride R and base address b:

- $f_{n,k}^{R,b}(X,.)$ provides for each lane, the index of the local register for the vector element accessed in the X^{th} slice.
- $g_{n,k}^{R,b,X}(I)$ provides the bank number of the vector element accessed by lane I in the Xth slice. Therefore $g_{n,k}^{R,b,X}$ must be a bijection on E^n to ensure conflict free access.
- for a given lane I, $h_{n,k}^{R,b,I}$ provides the successive indices in the local register for the elements accessed by the different slices: every index must be used one and only one time.

B. Proof of lemma 1

We will prove Lemma 1 by induction on k.

For k=0, Lemma 1 is equivalent to the Distribution Theorem for odd strided vectors on interleaved memories [1].

Let us now suppose that Lemma 1 is valid for any k smaller than K.

Let us note that if b and b' differ by a multiple of 2^k, one can use $f_{n,k}^{R,b}$ as $f_{n,k}^{R,b'}$ and vice-versa for any R,n and k.

We first give the sketch of the proof. The central part of the proof (i.e., the first step of function construction) is then detailed.

1) Proof sketch: The proof consists in constructing the function $f_{n,K}^{R,b}$ and its derived functions $g_{n,K}^{R,b,Y}$ and $h_{n,K}^{R,b,I}$ in two steps.

At a first step, we will construct $f_{n,K}^{R,b}(Y,I)$ for any $0 \leq Y < 2^{K-1}$ and for any $0 \leq I < 2^n$ with the two following properties:

Property 1: for any $0 \leq A < 2^{n-1}$, the unordered pair of reverse images for $\{2*A, 2*A+1\}$ by $g_{n,K-1}^{R,b,Y}$ becomes the undordered pair of reverse images for $\{A, A+2^{n-1}\}$ by $g_{n,K}^{R,b,Y}$.

Property 2: for any I in E^n, for any $0 \leq Y < 2^{K-1}$, $h_{n,K}^{R,b,I}(Y) = h_{n,K-1}^{R,b,I}(Y)$ or $h_{n,K}^{R,b,I}(Y) = h_{n,K-1}^{R,b,I}(Y) + 2^{K-1}$.

Property 1 ensures that $g_{n,K}^{R,b,Y}$ is therefore a bijection on E^n.

A function respecting these two properties is constructed in IV-B.2.

At a second step, for any $2^{K-1} \leq Y < 2^K$, for any I in E^n, we define $f_{n,K}^{R,b}(Y,I)$ by $f_{n,K}^{R,b}(Y,I) = f_{n,K}^{R,b}(Y - 2^{K-1}, I) \oplus 2^{K-1}$ where $\oplus$ is the exclusive OR. Then:

1) **The function $g_{n,K}^{R,b,Y}$ is bijection on E^n:**
for any I in E^n, $g_{n,K}^{R,b,Y}(I) = g_{n,K}^{R,b,Y-2^{K-1}}(I) \oplus 2^{n-1}$. As $g_{n,K}^{R,b,Y-2^{K-1}}$ is a bijection on E^n, $g_{n,K}^{R,b,Y}$ is also a bijection on E^n.
2) **The function $h_{n,K}^{R,b,I}$ is a bijection on E^K:**
Property 2 ensures that, for each integer X such that $0 \leq X < 2^{K-1}$, the undordered pair $\{h_{n,K}^{R,b,I}(X), h_{n,K}^{R,b,I}(X + 2^{K-1})\}$ is equal to the unordered pair $\{h_{n,K-1}^{R,b,I}(X), h_{n,K-1}^{R,b,I}(X) + 2^{K-1}\}$. As for any I in E^n, $h_{n,K-1}^{R,b,I}$ is a bijection on E^{K-1}, then $h_{n,K}^{R,b,I}$ is a bijection on E^K.

■

2) Details of the first step of $f_{n,K}^{R,b}$ construction: We detail here the first step of the construction of a function $f_{n,K}^{R,b}$ that respects Property 1 and Property 2

let us consider the fonction $G_{n,K}^{R,b,Y}$ defined by:
$$G_{n,K}^{R,b,Y} : \begin{array}{ccc} E^n & \rightarrow & E^n \\ i & \rightarrow & \frac{b+R*(i+2^n*f_{n,K-1}^{R,b}(Y,i))}{2^K} \ mod\ 2^n \end{array}$$
One can remark that, for every i, $\frac{g_{n,K-1}^{R,b,Y}(i)}{2} = G_{n,K}^{R,b,Y}(i) \ mod\ 2^{n-1}$, i.e., the $n-1$ least significant bits of $G_{n,K}^{R,b,Y}(i)$ are equal to the $n-1$ most significant bits of $g_{n,K-1}^{R,b,Y}(i)$.
let us consider an integer $0 \leq A < 2^{n-1}$,
as by hypothesis $g_{n,K-1}^{R,b,Y}$ is a bijection, there exists two integers I0, I1 in E^n such that $g_{n,K-1}^{R,b,Y}(I0) = 2*A$ and $g_{n,K-1}^{R,b,Y}(I1) = 2*A+1$.
Using the previous remark, we can deduce that

- $G_{n,K}^{R,b,Y}(I0) = A$ or $G_{n,K}^{R,b,Y}(I0) = A + 2^{n-1}$
- $G_{n,K}^{R,b,Y}(I1) = A$ or $G_{n,K}^{R,b,Y}(I1) = A + 2^{n-1}$

Let us distinguish two cases:

1) if $G_{n,K}^{R,b,Y}(I0)$ and $G_{n,K}^{R,b,Y}(I1)$ are different then we can fix:
$f_{n,K}^{R,b}(Y,I0) = f_{n,K-1}^{R,b}(Y,I0)$ and $f_{n,K}^{R,b}(Y,I1) = f_{n,K-1}^{R,b}(Y,I1)$ i.e., $g_{n,K}^{R,b,Y}(I0) = G_{n,K}^{R,b,Y}(I0)$ and $g_{n,K}^{R,b,Y}(I1) = G_{n,K}^{R,b,Y}(I1)$

2) if $G^{R,b,Y}_{n,K}(I0)$ and $G^{R,b,Y}_{n,K}(I1)$ are equal then we can fix:
$f^{R,b}_{n,K}(Y,I0) = f^{R,b}_{n,K-1}(Y,I0)$ and $f^{R,b}_{n,K}(Y,I1) = f^{R,b}_{n,K-1}(Y,I1) + 2^{K-1}$
i.e. $g^{R,b,Y}_{n,K}(I0) = G^{R,b,Y}_{n,K}(I0)$ and $g^{R,b,Y}_{n,K}(I1) = G^{R,b,Y}_{n,K}(I0) \oplus 2^{n-1}$

In both cases, the unordered pair (I0, I1), the reverse image of the unordered pair $\{2*A, 2*A+1\}$ by the function $g^{R,b,Y}_{n,K-1}$, is the reverse image of the unordered pair $\{A, A+2^{n-1}\}$ by the function $g^{R,b,Y}_{n,K}$: that is the defined function respects Property 1.

One can easily verify that, by construction, function $f^{R,b}_{n,K}$ also respects Property 2.

■

C. Proof of the theorem

First let us consider odd strided vectors.

For a vector V stored with an odd stride R and base address b, one can access the successive subslices S_j defined below:

at cycle j, Lane I accesses element $(I + 2^n f^{R,b}_{n,k}(j,I))\ mod\ 2^{n+k}$ of the vector.

Lemma 1 ensures that the acccess to any of the slices S_j is bank conflict free and that all elements of the vector are read.

More generally, Lemma 1 also allows to define conflict free accesses to subslices of any vector V stored with a stride of the form $2^r R$ with R odd and $r \leq k$ and base address as follows. For $j = j0 + 2^{k-r} j1$, slice S_j is defined by
Lane I accesses element $(I + 2^n (f^{R,B}_{n,k-r}(j0,I) * 2^r + j1))\ mod\ 2^{n+k}$ of the vector where $B = (b/2^r)$[1]

■

V. Practical use

Proof of Lemma 1 from the previous section provides a constructive method to compute the the indices of elements to be accessed in parallel for any strided vector with stride $2^r R$, $r \leq k$ by each lane. We have implemented this algorithm as a C program.

However, this method cannot be implemented as a one-cycle hardware automaton in a processor. But the set of indexes for each possible (stride, base address) pair can be precomputed and stored in a small ROM associated with each lane. We analyze here the size of this ROM and show that this size remains limited for practical cases as in Tarantula [2].

[1]The formula in the version of the paper published in IEEE TC was incorrect, thanks to Chunyang Gou from TU Delft for pointing us this error

A. Information needed for odd-strided vectors

R being the odd stride of vector V and b being its base address, the function $f^{R,b}_{n,k}$ constructed in Lemma 1 proof only depends on the k least significant bits of b (see section 4.2) and on the $n+k$ least significant bits of R. Since R is odd, its least significant bit of R is always equal to one.

Therefore, only 2^{n+2k-1} different functions $f^{R,b}_{n,k}$ are used. For a given lane I, 2^k words of k bits (each one corresponding to the index of the vector element to be accessed on a given subslice) must be memorized per function. However by using the symmetry in the access slices (2nd half can be deduced from first half by flipping the most significant bit), one can store only 2^{k-1} words of k bits per function on each lane.

In summary, 2^{n+3k-2} words of k bits are needed to store all the indexes related to odd strided vectors.

The index of the local vector element accessed by the lane in slice X is obtained from word $((R/2)\ mod\ 2^{n+k-1}) * 2^{2k-1}) + (b\ mod\ 2^k) * 2^{k-1} + (X\ mod 2^{k-1})$ and then flipping the most significant bit of this value if X is higher than 2^{k-1}.

B. Information needed for accessing other strided vectors

Note that by construction (see Section IV), $f^{R,b}_{n,k-r}$ and $f^{R,b}_{n,k}$ coincidate for any input on the last $k-r$ bits. Therefore, the slice information for an even stride $2^r R$ ($0 < r \leq k$) can be deduced at run time from the information for stride R by simply deleting the r most significant bits.

The index of the local vector element accessed by the lane in slice $X = X0 + 2^r\ X1$ is obtained as follows:

1) Value v of word $((R/2)\ mod\ 2^{n+k-1}) * 2^{2k-1}) + (B\ mod\ 2^k) * 2^{k-1} + (2^r * X1\ mod\ 2^{k-1})$ is read on the ROM ($B = (b/2^r)$
2) the index is computed as $(2^r * v + X0)\ mod\ 2^k$.

C. Tarantula processor

The Tarantula processor features 16 lanes and a cache line of 8 words, i.e., $n = 4$ and $k = 3$. Therefore, in pratice one has to store only 2048 words of 3 bits per lane, i.e, 768 bytes in a ROM, to be able to compute a conflict free subslice distribution for any vector stored with a stride of the form $2^r R$ ($0 < r \leq 3$), R odd.

As mentioned in [2], the address generator in each lane must also feature a 7x64 bit multiplier in order to generate the addresses of the vector elements. For a read, the sequencing of a memory access is as follows: 1) acquisition of the local number of the subslice element 2) computation of the vector element address 3) virtual-physical address translation 4) routing to the memory banks,5) memory bank access 6) routing back to the lanes.

One will note that no extra buffering of addresses and/or data is needed on this path.

VI. Conclusion

The implementation of a single die vector microprocessor has become possible as illustrated by the Tarantula project [2]. The bank interleaved L2 cache is the source of operands for the vector unit. The parallel access to strided vectors in cache is therefore one of the critical issue for performance on vector applications. Unfortunately the conventional conflict free access to parallel slices of consecutive elements of a vector implies the use of single word cache block size, thus leading to unacceptable implementation cost for the tag array. On the other hand, multiword cache block size is highly desirable on a vector microprocessor, since it allows to exploit spatial locality at a modest tag storage cost.

In this paper we have exhibited a solution for using a multiword cache block size while exploiting the full cache bandwidth on a very large family of strided vectors. We leverage the two degrees of vector parallelism (lanes and vector registers). Elements of a vector are accessed out-of-order, but at a rate of one element per lane and per cycle.

The theory developed in this paper was used for the design of the Tarantula project [2].

References

[1] P. Budnik and D. Kuck. The organization and use of parallel memories. *IEEE Trans. Computers*, C-20:1566–1569, December 1971.

[2] Roger Espasa, Federico Ardanaz, Joel Emer, Stephen Felix, Julio Gago, Roger Gramunt, Isaac Hernandez, Toni Juan, Geoff Lowney, Matt Mattina, and André Seznec. Tarantula: A vector extension to the alpha architecture. In *Proceedings of the 29th International Symposium on Computer Architecture*, May 2002.

[3] D. Harper and J. Jump. Vector access performance in parallel memories using A skewed storage scheme. *IEEE Transactions on Computers*, C-36(12):1440–1449, December 1987.

[4] Duncan H. Lawrie and Chandra R. Vora. The prime memory system for array access. *IEEE Transactions on Computers*, C-31(5):435–442, May 1982.

[5] Montse Peiron, Mateo Valero, Eduard Ayguade, and Tomas Lang. Vector multiprocessors with arbitrated memory access. In *Proc. of the 22nd Annual International Symposium on Computer Architecture*, June 1995.

[6] André Seznec and Jacques Lenfant. Interleaved parallel schemes: Improving memory throughput on supercomputers. In *Proceedings the 19th Annual International Symposium on Computer Architecture*, May 1992.