PAPER

MVP-Cache: A Multi-Banked Cache Memory for Energy-Efficient Vector Processing of Multimedia Applications

Ye GAO[†a)], **Masayuki SATO**[†,††b)], ***Nonmembers***, **Ryusuke EGAWA**[†,††c)], **Hiroyuki TAKIZAWA**[†††,††d)], ***and*** **Hiroaki KOBAYASHI**[†,††e)], ***Members***

SUMMARY Vector processors have significant advantages for next generation multimedia applications (MMAs). One of the advantages is that vector processors can achieve high data transfer performance by using a high bandwidth memory sub-system, resulting in a high sustained computing performance. However, the high bandwidth memory sub-system usually leads to enormous costs in terms of chip area, power and energy consumption. These costs are too expensive for commodity computer systems, which are the main execution platform of MMAs. This paper proposes a new multi-banked cache memory for commodity computer systems called MVP-cache in order to expand the potential of vector architectures on MMAs. Unlike conventional multi-banked cache memories, which employ one tag array and one data array in a sub-cache, MVP-cache associates one tag array with multiple independent data arrays of small-sized cache lines. In this way, MVP-cache realizes less static power consumption on its tag arrays. MVP-cache can also achieve high efficiency on short vector data transfers because the flexibility of data transfers can be improved by independently controlling the data transfers of each data array.

key words: *vector architecture, multimedia application, multi-banked cache memory*

1. Introduction

Media processors are required to achieve a higher performance because their target applications, multimedia applications (MMAs), will process an unprecedented amount of data in the future in order to improve the quality of media services. For example, a next generation video application, 8K video, needs to process 64× more data than current standard definition videos [1], and a 3D computer vision application for super resolution images [2] needs to process 25 to 200× more data than current ones.

In order to efficiently process the large amount of media data, the vector architecture [3] becomes one of the best candidates for the design of high performance media processors. This is because the vector architecture could efficiently compute and transfer the large amount of media data by exploiting data level parallelism involved in MMAs. MMAs usually contain packs of independent data that have the same operations, which can be processed in parallel [4]. The pack is called a *vector*, and the number of elements in a vector is called *vector length* in this paper. The vector architecture can process each vector by one instruction, and thus their hardware resources such as parallelized functional units and cache ports can be kept busy as continuously as possible. As a result, the vector architecture can potentially achieve high computing performance for MMAs.

Modern vector architectures usually employ a multi-banked cache memory in order to improve their data transfer performance [5]–[8]. The memory subsystem with multi-banked cache memory can provide data to parallelized functional units at a sufficient transfer rate. Therefore, the vector architectures could achieve high sustained computational performance. Since some previous researches have shown that MMAs have high data reusability [9], [10], the multi-banked cache memories also have a high potential for MMAs.

However, conventional multi-banked cache memories of vector processors cannot satisfy the design requirements of MMAs. There are at least two requirements that should be considered when designing a multi-banked cache memory for MMAs. One is that the multi-banked cache memory should efficiently transfer vectors of various lengths because MMAs contains not only long vectors but also short vectors. The other is that the multi-banked cache memory should achieve low energy consumption. This is because, as the main execution environment of MMAs, commodity computer systems cannot invest so much energy consumption for a cache memory. However, conventional multi-banked cache memories of vector processors cannot satisfy both of the requirements at the same time. They either have low data transfer performance for short vectors or cost high energy consumption. Therefore, a multi-banked cache memory needs to improve its data transfer performance for short vectors at low energy consumption.

To this end, this paper proposes a multi-banked cache memory for vector processors called MVP-cache. Unlike conventional multi-banked cache memories that consist of one data array and one tag array in each bank, MVP-cache associates one tag array with multiple independent data arrays of small-sized cache lines. In this way, MVP-cache could reduce its energy consumption by decreasing the number of tag arrays. At the same time, MVP-cache can also

Manuscript received July 3, 2014.
Manuscript publicized August 22, 2014.

[†]The authors are with Cyberscience Center, Tohoku University, Sendai-shi, 980–8578 Japan.
[††]The authors are with JST CREST, Kawaguchi-shi, 332–0012 Japan.
[†††]The author is with the Graduate School of Information Sciences, Tohoku University, Sendai-shi, 980–8578 Japan.
a) E-mail: gaoye@sc.isc.tohoku.ac.jp
b) E-mail: masayuki@sc.isc.tohoku.ac.jp
c) E-mail: egawa@isc.tohoku.ac.jp
d) E-mail: tacky@isc.tohoku.ac.jp
e) E-mail: koba@isc.tohoku.ac.jp
DOI: 10.1587/transinf.2014EDP7227

achieve high performance on short vector data transfers because the flexibility of data transfers could be improved so that all the data arrays can fully utilize their bandwidth for transferring short vectors.

The rest of the paper is organized as follows. Section 2 clarifies the challenges for designing a multi-banked cache memory for MMAs. Section 3 describes the details of MVP-cache, and Sect. 4 evaluates the performance of MVP-cache. Section 5 gives the conclusions of this paper.

2. Challenges for Designing a Multi-Banked Cache Memory for MMAs

This section aims to clarify the drawbacks of conventional multi-banked cache memories in order to show challenges for designing a multi-banked cache memory for MMAs.

Multi-banked cache memories have a high potential to improve data transfer performance. A multi-banked cache memory typically consists of multiple independent cache banks, called sub-caches, and each sub-cache is connected to cache ports of the vector unit via interconnection fabric. The vector unit is the hardware unit that is in charge of vector processing in the vector architecture. When a cache hit occurs, data can be simultaneously transferred from/to multiple independent sub-caches. In this way, the data transfer performance could be improved in comparison with a single bank cache memory. Although the multi-banked cache memories bring additional energy consumption for vector processors, it is smaller than the energy saving induced by reducing the number of off-chip memory accesses.

Conventional multi-banked cache memories can be classified into the multi-banked cache memories with large cache lines (MBC-L) [11] and with small cache lines (MBC-S) [7]. In this paper, a multi-banked cache memory is classified into MBC-S if its cache line size is 8 bytes or smaller, otherwise it is classified into MBC-L. The borderline between MBC-S and MBC-L is set at 8 bytes because it is the size of one double-precision floating-point data, which is one of the most common data type for vector processors.

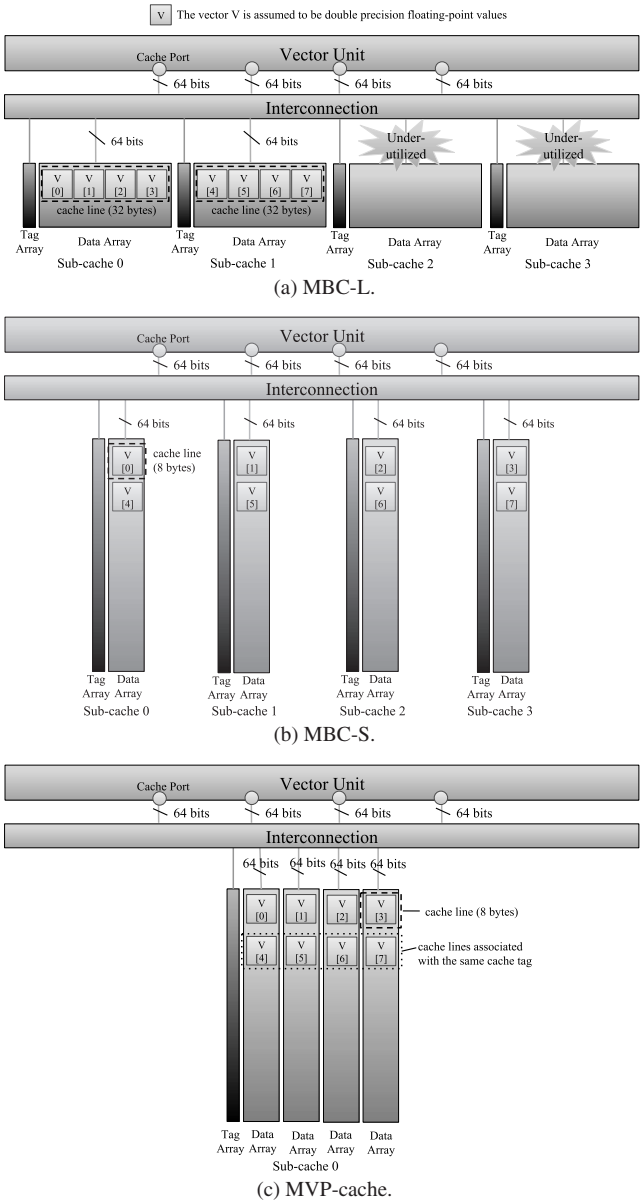

Fig. 1 Short vector data transfer in MBC-L, MBC-S and MVP-cache.

2.1 MBC-L

MBC-L allocates several data elements with consecutive addresses in the same sub-cache. Let V be a vector of eight double-precision floating-point elements. Then, Fig. 1 (a) shows an example of MBC-L and the data layout of V. In this example, there are four sub-caches, and each of them can transfer 64-bit data per cycle. The cache line size of this MBC-L is assumed to be 32 bytes. In this case, since each cache line can store four elements, the consecutive elements $V[0]$ to $V[3]$ are stored in *sub-cache*0, and elements $V[4]$ to $V[7]$ are stored in *sub-cache*1.

The drawback of MBC-L is that MMAs with short vectors cannot make a good use of its high cache bandwidth. Figure 1 (a) illustrates the drawback of using MBC-L for MMAs with short vectors. The vector V only has eight elements. Those eight elements are stored in the cache lines of *sub-cache*0 and *sub-cache*1, and there are no data stored in the cache lines of *sub-cache*2 and *sub-cache*3. Therefore, the data can be only transferred from *sub-cache*0 and *sub-cache*1 in parallel. Since two of four sub-caches are underutilized, the sustained bandwidth only reaches half of the peak bandwidth.

In practical uses, the underutilization of sub-caches due to short vectors will be more serious than the example in Fig. 1 (a), where V is assumed to be a vector of double-precision floating-point values. In a realistic situation, single-precision floating-point values or integer values are also commonly used in MMAs. Since the size of these data is smaller than that of double-precision floating-point values, the number of vector elements that are stored together in one sub-cache increases. This leads to even lower utilization than that in the case of double-precision floating-

point values. Moreover, to obtain a high cache bandwidth, a large number of sub-caches would be employed. Espasa et al. [7] have proposed an MBC-L for a vector ISA extension. Their MBC-L adopts 16 sub-caches and the cache line size of each sub-cache is 64 bytes, which is the size of eight double-precision floating-point elements. This configuration means that, if the length of a double-precision floating-point vector is shorter than 128, its data transfer potential would be underutilized.

2.2 MBC-S

In contrast to MBC-L, MBC-S is able to efficiently transfer short vector data. Figure 1 (b) illustrates a data layout in the case of MBC-S with 8-byte cache lines. Since each cache line of MBC-S can only store one double-precision floating-point data, the consecutive elements V[0] to V[3] are dispersed in *sub-cache*0 to *sub-cache*3, respectively. By making full use the sub-caches, MBC-S is able to transfer short vector data at a high bandwidth.

On the other hand, MBC-S consumes more energy consumption than MBC-L because the tag array size of MBC-S is larger than that of MBC-L if they have the same capacity. MBC-S has to employ larger tag arrays because its small-sized cache lines lead to a large number of cache sets. Therefore, the reduction in energy consumption of tag arrays should be considered when designing a multi-banked cache memory for MMAs. Although our previous research [12] has proposed several techniques to reduce the energy consumption of cache memories, it only reduces the energy consumption of data arrays.

2.3 Related Work

Espasa et al. have proposed an out-of-order vector architecture [13] and have applied it to Tarantula [7], a vector extension to the Alpha processor. The cache mechanism of Tarantula, which is categorized into MBC-L, employs 16 sub-caches and an address re-ordering algorithm [14] to reduce the bank conflictions. However, performance degradation of MMAs with short vectors could occur due to inefficient short vector transfers of MBC-L.

Batten et al. have noted that not only the access latency of memory sub-systems but also their bandwidth is very important to improve the application performance [8]. They have proposed an inexpensive non-blocking cache memory for vector architectures to improve the bandwidth and reduce the access latency of memory sub-systems. Musa et al. have designed a vector cache for vector architectures [11]. The vector cache introduces a bypass mechanism and miss status handling registers to improve the sustain memory bandwidth to next generation vector supercomputers. However, both of the two proposals adopt MBC-S and cause high energy consumption in the tag arrays.

To sum up the above discussion, when considering the design of a media-oriented processor for commodity computer systems, neither MBC-L nor MBC-S can satisfy requirements of high data transfer performance at low energy consumption. MBC-L cannot transfer short vectors efficiently, and MBC-S consumes a large energy on tag arrays. Therefore, a new organization of a multi-banked cache memory is needed to overcome the drawbacks of MBC-L and MBC-S.

3. MVP-Cache

3.1 Key Idea and Potential of MVP-Cache

MVP-cache aims to achieve high data transfer performance for short vectors at low energy consumption. Essentially, MBC-L consumes low energy on tag arrays because it associates one cache tag with a large-sized cache line. Meanwhile, MBC-S is efficient on short vector data transfer because consecutive vector elements are dispersed across different sub-caches due to the small-sized cache line. Therefore, in order to obtain the advantages of both MBC-L and MBC-S, MVP-cache associates one tag array with multiple independent data arrays that employ the small-sized cache lines. As a result, MVP-cache could reduce the size of tag arrays compared with MBC-S. At the same time, the independent small-sized cache lines of MVP-cache make it possible to tolerate even shorter vectors than MBC-L. Therefore, MVP-cache could potentially satisfy the requirements of high data transfer performance for short vectors at low energy consumption.

Figure 1 (c) shows the organization of MVP-cache. It is assumed that MVP-cache has the same capacity as MBC-L and MBC-S shown in Figs. 1 (a) and 1 (b), respectively. It is also supposed that MVP-cache associates one tag array with four data arrays, each of which adopts 8-byte cache lines. Compared with MBC-S, the size of tag array of MVP-cache is one-fourth of that of MBC-S because MBC-S associates one tag array with one data array.

Moreover, Fig. 1 (c) also shows the data layout in MVP-cache. The consecutive elements V[0] to V[3] can be dispersed in the different independent data arrays because each cache line in data arrays can only store one double-precision floating-point element. In this way, MVP-cache could transfer short vectors by fully using the bandwidth, while MBC-L can only use a half of the bandwidth as mentioned in Sect. 2.1. Therefore, MVP-cache is potentially more efficient for MMAs with short vectors than MBC-L.

3.2 Organization of MVP-Cache

Figure 2 shows a block diagram of MVP-cache. As mentioned in Sect. 3.1, MVP-cache associates one tag array with multiple data arrays consisting of small-sized cache lines in a sub-cache. Each of data arrays adopts the cache line size of 8 bytes, which is the size of a double-precision floating-point value, and is independently connected to cache ports of the vector unit via a crossbar. The cache lines that are associated with the same tag act as an atomic unit of data management and miss handling. If a cache hit occurs, cache

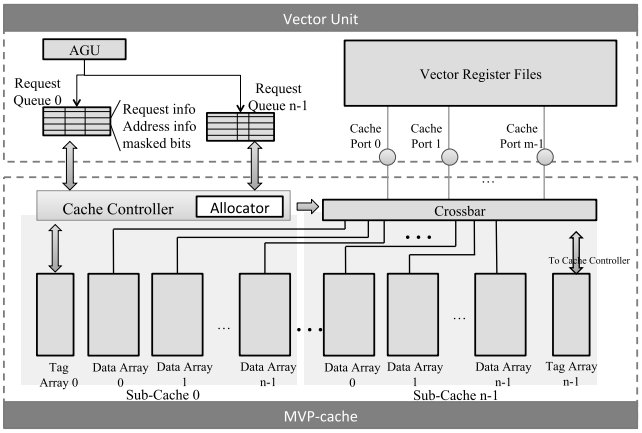

Fig. 2 Block diagram of MVP-cache.

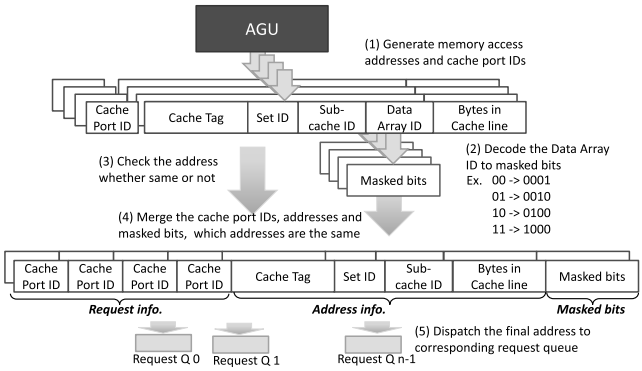

Fig. 3 Generation of *Request Info*, *Address Info* and *Mask Bits*.

lines that are associated with the same cache tag will be transferred.

Unlike conventional cache memories, MVP-cache associates one cache tag with multiple cache lines, which are located in different data arrays. However, sometimes not all these cache lines are required to transfer their data, such as in the situations of unaligned data accesses, stride accesses and indexed memory accesses. Therefore, it is necessary to ascertain which cache lines are required. MVP-cache realizes such a kind of control by adding a new field in a cache request, called *Mask Bits*. If a mask bit is set to one, the cache line should be transferred from the corresponding data array to the vector unit, otherwise the data transfer is disabled. In this way, MVP-cache avoids transferring the unnecessary data and improves its energy efficiency.

3.3 Data Transfer Control Information of MVP-Cache

In order to control the data transfer of MVP-cache, its cache controller needs some cache access information including *Request Info*, *Address Info* and *Mask Bits*. *Request info* stores information of the cache port IDs requested by a cache access. *Address Info* is the target address of the cache access. *Mask Bits* own the hints that which data arrays in a sub-cache should carry out data transfers. *Address Info* is used to judge if a cache access hits or misses, while *Request Info* and *Mask Bits* are used by the crossbar allocator to generate the connections between MVP-cache and cache ports of the vector unit.

The cache access information is generated according to memory addresses and stored in the data request queues. Steps to generate *Request Info*, *Address Info* and *Mask Bits* are shown in Fig. 3 and described as follows.

Step 1 Generate the memory addresses accessed by a vector load/store instruction, and determine the cache port IDs used for the accesses. Each memory address contains fields of `cache tag`, `set ID`, `sub-cache ID`, `data array ID` and `bytes in a cache line`. These fields can be obtained by using logical SHIFT and bit-wise AND operations. For example, a data array ID can be obtained as follows. Suppose that the number of data arrays in each sub-cache is 2^N. The logical SHIFT operation is used to remove the least significant three bits because the cache line size is 8 bytes. Then, the bit-wise AND operation is used to retrieve the least significant N bits from the shifted value that is the data array ID of a given memory address. The other fields can also be calculated from a memory address.

Step 2 Decode `data array IDs` to set mask bits. If the `data array ID` of a memory access is n, the n th mask bit should be set as one, because the `data array ID` means that the specified data array owns the data that should be transferred.

Step 3 Check whether there are the same memory addresses in the generated addresses. The `cache tag`, `set ID` and `sub-cache ID` of all access addresses are compared each other. If they are the same, it means that they would access the same set of the data arrays in the same sub-cache.

Step 4 If there are the same memory addresses, their `cache port IDs`, memory addresses, and `mask bits` are merged to generate *Request Info*, *Address Info*, and *Mask Bits*, respectively. For *Mask Bits*, they are merged by using the bit-wise OR operation with the other mask bits. For `cache port IDs`, they are merged by using logical SHIFT and bit-wise AND operations to hold the whole information.

Step5 Dispatch the final results to the corresponding request queues according to the `Sub-cache ID`.

The MVP-cache controls data transfers by using mask bits. This means that MVP-cache requires more complicated control than MBC-S and MBC-L. The complicated control potentially leads to a longer cache access latency and a larger area of control logics. Regarding the longer access latency, it has been shown in [15] that the long access latency of MVP-cache could be effectively hidden by using the out-of-order vector processing mechanism. Moreover, regarding the area of control logics, the overhead is extremely small. The additional hardware of MVP-cache on control logics is for the buffer to store the mask bit. We suppose that the buffer contains 128 entries, which corresponds

to entries of the vector memory instruction buffer. We also suppose that the number of data arrays is 128, which is the maximum number used in the evaluation. Since the number of mask bits equals the number of data array, the buffer size equals 2KB (128 bits × 128 entries). Compared with 4MB cache capacity used in the evaluation, the buffer only takes 0.05% extra area.

3.4 MVP-Cache Crossbar Allocator

The MVP-cache controller adopts an $n \times m$ allocator to control the crossbar, where n is the number of data arrays of all sub-caches, and m is the number of cache ports. The base design of the allocator and crossbar is described in [16], [17]. The input of the allocator is a request matrix $R = (r_{i,j})_{n\times m}$, where the element $r_{i,j} = 1$ means that data array i needs to use cache port j. The output of the allocator is a grant matrix $G = (g_{i,j})_{n\times m}$, where the element $g_{i,j} = 1$ means that data array i is allowed to use cache port j. $G = (g_{i,j})_{n\times m}$ is used to configure the crossbar of MVP-cache.

MVP-cache uses *Mask Bits* to control which data array transfers the data to the vector unit. Therefore, it is also necessary to design an algorithm that generates the request matrix $R = (r_{i,j})_{n\times m}$ with the concern of *Mask Bits*. The algorithm is shown in Algorithm 1. The $globalMaskBit[i]$ and $globalRequestInfo[i]$ mean the *Mask Bits* and *Request Info* of all sub-caches, respectively. They are generated by combining the local *Mask Bits* and *Request Info* of each sub-caches together. By using Algorithm 1, *Mask Bits* can be used to control data transfers that are really necessary for the vector unit as mentioned in Sect. 3.2.

3.5 Tag Array Conflicts of MVP-Cache

A situation of memory accesses to the different sets in the same sub-cache is called a tag array conflict in this paper. The tag array conflicts lead to performance degradation because a sub-cache can judge a cache hit or miss only once per cycle. Tag array conflicts frequently occur in the case of stride accesses. The stride access is a kind of accesses to every k-th vector element, where k should be bigger than two. For example, the accesses to $V(0)$, $V(2)$, $V(4)$ ... are a stride access whose stride length is two. A longer stride length may increase the probability that vector elements to be accessed are located in the different sets in the same sub-cache. Therefore, the longer the stride length is, the more tag array conflicts occur.

In this paper, we tune two parameters of MVP-cache in order to reduce the tag array conflicts in stride accesses. One is the numbers of sub-caches. The other is the numbers of data arrays in one sub-cache. Increasing the number of sub-caches leads to a higher probability that memory accesses are dispersed across different sub-caches, while increasing the number of data arrays increases the probability that memory accesses are concentrated on the same set in the same sub-cache. However, the increase in the numbers of data arrays or sub-caches also brings extra energy consumption. A large number of data arrays would lead to high power consumption on the crossbar between data arrays and cache ports of the vector unit. Meanwhile, increasing the number of sub-caches would lead to high the energy consumption of tag arrays. Therefore, this paper evaluates various combinations of the numbers of data arrays and sub-caches in order to find an energy-efficient configuration of MVP-cache for MMAs.

Algorithm 1 Generation of Request Matrix $R = (r_{i,j})_{n\times m}$ Using Mask Bits and Request Info

```
for i = 0 → n − 1 do
    for j = 0 → m − 1 do
        if globalMaskBit[i] == 1 and j == globalRequestInfo[i] then
            r[i][j] ← 1
        else
            r[i][j] ← 0
        end if
    end for
end for
```

4. Performance Evaluations

4.1 Experimental Methodology

In order to evaluate MVP-cache, a simulator of the vector processor shown in Fig. 4 is developed based on the SimpleScalar toolset [18]. The original SimpleScalar is used for the scalar unit of the vector processor. In addition, an out-of-order vector processing mechanism (OVPM) and MVP-cache are implemented in SimpleScalar. OVPM has been proposed in [15] to efficiently process short vectors and hide the long access latency of MVP-cache. During the execution, if a vector instruction is identified in the decode stage, it is dispatched to the vector memory instruction buffer or the vector arithmetic instruction buffer. The vector instructions in the two instruction buffers are passed to the corresponding vector issue queues as long as their operands are

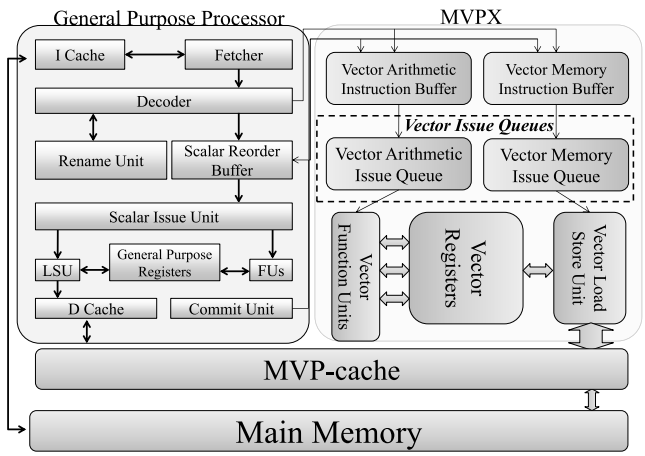

Fig. 4 Baseline processor used in the evaluation.

Table 1 Configuration of MVP and memory sub-system.

Parameters	Value
Process Technology	32 nm
Processor Frequency	1GHz
Vector ALU Pipeline Latency	10 cycles
Vector Multiplier Pipeline Latency	15 cycles
Vector Division Pipeline Latency	20 cycles
Number of Vector Lanes	8
Number of Architectural Vector Registers	16
Number of Physical Vector Registers	96
Entries per Vector Register	128 entries
The size of VMIB and VAIB	128 entries
The size of Vector Load and Store Queue	512 entries
Number of Cache Ports	8
Cache Capacity	4 MB
Cache Bandwidth	64 bytes/cycle
Cache Line Size of MVP-cache	8 bytes
MVP-cache Access Latency	20 cycles
Number of Sub-caches	4
Number of Data Arrays in a Sub-cache	8
Memory Bandwidth	32 bytes/cycle
Main Memory Latency	100 cycles
Cache Line Size of MBC-S	8 bytes
Cache Line Size of MBC-L	64 bytes
MBC-L Access Latency	10 cycles
MBC-S Access Latency	10 cycles
Number of Sub-caches of MBC-S and MBC-L	32

ready. In the vector memory issue queue, the instructions wait for the availability of the vector load and store unit, which consists of an address generation unit, cache ports, and request queues. If the vector load and store unit becomes available, a vector memory instruction is issued to the unit to generate the memory address and access MVP-cache. At last, the instruction is passed to the commit unit and waits to be committed. Therefore, the decode function, issue function, writeback function and commit function of SimpleScalar are extended to process vector instructions as mentioned above.

McPAT0.8 [19] and CACTI6.5 [20] are also used to evaluate the energy consumption of MVP-cache. The OVPM specification of MVP-cache MBC-S, and MBC-L are listed in Table 1. The cache line sizes of MVP-cache, MBC-S, and MBC-L are 8 bytes, 8 bytes and 64 bytes, respectively. To fairly compare performances of the three kinds of cache memories, their cache capacities and cache bandwidths are set to the same values. All the three multi-banked memories support the interleaved memory accesses to hide the access latencies and avoid bank conflicts.

Nine multimedia benchmark programs in Table 2 are used to evaluate MVP-cache. The benchmark programs of *clip*, *fft* and *power* are three hot kernels of the 3-D computer vision algorithm for super resolution images [2]. $M\times M$ and $V\times M$ are programs of matrix multiplications, which are commonly used in MMAs. The other four benchmark programs are the kernels of MMAs selected from the PARSEC benchmark suite [21] and the ALPbench benchmark suite [22]. Both of them include emerging MMAs that contain massive data level parallelism. Among the nine multimedia benchmark programs, only sphinx contains stride accesses. The typical stride length of *sphinx* is 13. The benchmark programs are compiled by the PISA cross-compiler, so as to generate assembly codes defined in the SimpleScalar toolset. For the vector codes, there are many mature vector compilers such as NEC's C compiler of SX-9 (sxcc) [23]. However, since SimpleScalar does not support the cross-compilation with sxcc, we firstly vectorize the benchmark programs automatically by using sxcc. Then, referring to the assembly code generated by sxcc, vector instructions are manually inserted into the assembly code files generated by the PISA cross-compiler. At last, the assembly code files are translated to binary codes as the inputs of the simulator.

Table 2 Benchmark programs.

Benchmarks	Categories	Vector Length
sphinx	speech recognition	4096
face	face recognition	173
ray	animation	1080
vips	image processing	79
clip	computer vision	64
fft	computer vision	32
power	computer vision	33
$M\times M$	Matrix Multiplication	1000
$V\times M$	Vector-Matrix Multiplication	1000

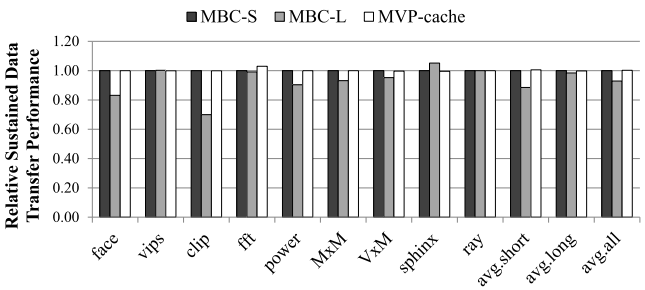

Fig. 5 Data transfer performance of MVP-cache.

To take a clear look at evaluation results, the benchmark programs are classified into the short vector category and the long vector category. The short vector category contains the benchmark programs *face*, *vips*, *clip*, *fft*, and *power*. The other benchmark programs including *ray*, $V\times M$, $M\times M$ and *sphinx* belong to the long vector category.

4.2 Evaluation Results

The purpose of MVP-cache is to achieve high data transfer performance, especially for short vector data transfer, at low energy consumption. Therefore, the data transfer performance and energy consumption of MVP-cache are compared with those of MBC-S and MBC-L to confirm its effectiveness. Then, the design space of MVP-cache is explored in order to find the energy efficient configuration for MMAs.

4.2.1 Data Transfer Performance of MVP-Cache

Figure 5 compares sustained bandwidths of MBC-S, MBC-L and MVP-cache, which are normalized by the bandwidth of MBC-S. MVP-cache attains almost the same performance as MBC-S, and higher performance than MBC-

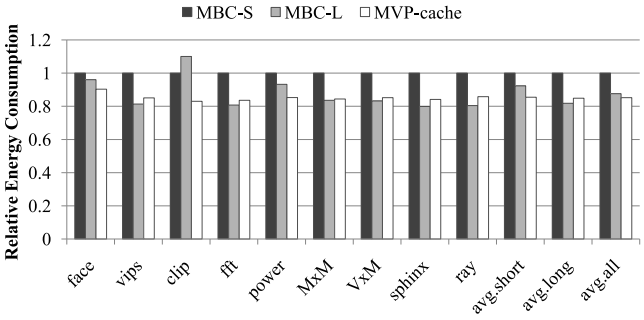

Fig. 6 Comparison of energy reduction among MBC-S, MBC-L and MVP-cache.

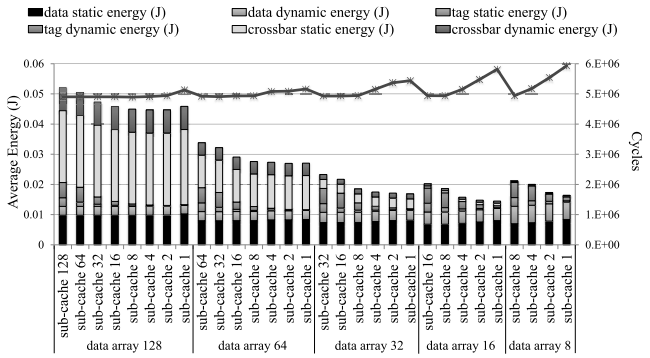

Fig. 7 Average energy consumption and execution cycles of all benchmark programs at various configuration of MVP-cache.

L. The performance improvement against MBC-L is significant for MMAs in the short vector category because of the efficient use of the bandwidths of all sub-caches. The benchmark program, *vips*, is an exception to MMAs in the short vector category because there is no data reusability in this benchmark program. However, since most of MMAs have high data reusability, MVP-cache could effectively achieve high data transfer performance for MMAs with short vector.

4.2.2 Energy Reduction by MVP-Cache

Figure 6 shows the relative energy consumptions of MVP-cache, MBC-S and MBC-L, which are normalized by the energy consumption of MBC-S. Although MVP-cache achieves almost the same performance as MBC-S as shown in Fig. 5, it consumes 15% less energy than MBC-S. This is because MVP-cache consumes a less power on tag arrays by associating one tag array with multiple data arrays. Meanwhile, MVP-cache attains 10% performance improvement but only achieves 3.7% energy reduction against MBC-L. This is because MVP-cache consumes higher power than MBC-L due to the complex crossbar.

Specially, MVP-cache achieves lower energy consumption than MBC-S and MBC-L for MMAs in the short vector category. This is due to the energy reduction in tag arrays against MBC-S and the performance improvement in short vector data transfers against MBC-L. On the other hand, for MMAs in the long vector category, MVP-cache achieves lower energy consumption than MBC-S but slightly higher energy consumption than MBC-L because MBC-L can also transfer long vectors at a high bandwidth as well as MVP-cache. Since the energy reduction effect of MVP-cache in the short vector category is larger than its energy consumption overhead in the long vector category, MVP-cache obtains the lowest energy consumption of average. Therefore, these evaluation results show that MVP-cache is an energy efficient approach for MMAs.

4.2.3 Discussion on MVP-Cache Configurations

As mentioned in Sect. 3.5, tag array conflicts will occur and degrade the performance in the case of stride accesses. For example, in the benchmark programs, *sphinx* contains long stride accesses. The tag array conflicts could be reduced by increasing the number of sub-caches and the number of data arrays that are associated with one tag array. However, the increase in hardware also leads to high energy consumption. Therefore, in order to find the most energy-efficient configuration of MVP-cache, this evaluation clarifies the trade-off between the reduction in tag array conflicts and the increase in energy consumption when adjusting the numbers of data arrays and sub-caches.

Figure 7 shows the average energy consumption breakdowns and execution cycles of all benchmark programs with changes of the numbers of sub-caches and data arrays. In the figure, n (n = 1, 2, 4, 8, 16, 32, 64, 128) and m (m = 8, 16, 32, 64, 128) represent the numbers of sub-caches and data arrays, respectively. Dividing n by m leads to the number of data arrays that are associated with one tag array. Further, when changing the number of data arrays, the total capacity of MVP-cache is kept constant by adjusting the number of sets in one sub-cache.

As shown in Fig. 7, MVP-cache achieves the lowest energy consumption in the case of one tag array associated with 16 data arrays. There are two reasons why such a configuration shows the lowest energy consumption. One is that, although a larger number of data arrays could reduce the number of tag array conflicts, the increase in the energy consumption of the crossbar is also significantly large especially when the number of data arrays is bigger than 16. The other is that MVP-cache of eight data arrays consumes a larger dynamic energy on data arrays than that of 16 data arrays. This is because MVP-cache of eight data arrays contains a larger number of cache sets in one sub-cache, which leads to a higher driving energy for each cache access.

Meanwhile, when the number of data arrays is fixed to 16, the energy consumption of MVP-cache reduces with the decrease in the number of sub-caches because of less tag arrays. Although a smaller number of sub-caches increase the frequency of tag array conflicts occurred in *sphinx*, the increase in execution cycles is smaller than the reduction in energy consumption. Therefore, one tag array associated with 16 data arrays is the most energy efficient configuration for the MMAs used in this evaluation. We believe that the proposed technique can also find a reasonable configuration

Table 3 Lowest-energy configuration for different stride length.

Stride Length	Lowest-energy Configuration
Stride 1	data array 16, sub-cache 1
Stride 2	data array 16, sub-cache 1
Stride 3	data array 16, sub-cache 1
Stride 4	data array 32, sub-cache 1
Stride 5	data array 32, sub-cache 1
Stride 6	data array 32, sub-cache 1
Stride 7	data array 32, sub-cache 8
Stride 8	data array 32, sub-cache 8

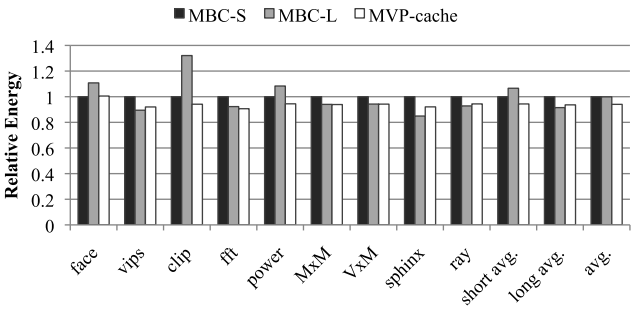

Fig. 8 Total energy consumption of a processor.

of MVP-cache for other MMAs not evaluated in this work.

4.2.4 Energy Consumption for Different Stride Lengths

In order to investigate energy consumptions in the cases of different stride lengths, we evaluate MVP-cache by using some micro benchmark programs. They are designed as an addition of two vectors with different stride lengths. Table 3 shows the lowest-energy configurations of MVP-cache when changing of the stride length. The results are normalized against the energy consumption in the case of 128 data arrays and 128 sub-caches.

With the increase in the stride length, the impacts from stride accesses would also increase. Hence, if the stride length ranges from one to eight, the lowest energy configurations of MVP-cache are different. The configurations of 16 data arrays sharing one tag array, 32 data arrays sharing one tag array and 32 data arrays sharing four tag arrays achieve the lowest energy consumptions for the stride lengths ranging from 1 to 3, 4 to 6, and 7 to 8, respectively. On average, these three configurations achieve almost the same energy consumption. However, since the stride lengths of most of MMAs are usually less than three [14], the configuration of 16 data arrays sharing one tag array could be considered the most efficient configuration for MMAs. These results are also in accord with the result in Sect. 4.2.3.

4.2.5 Total Energy Consumption of Processor

Figure 8 shows the total energy consumptions of vector processors with MBC-S, MBC-L and MVP-cache. The energy consumption of a vector processor with MVP-cache is 6% lower than those of vector processors with MBC-S and MBC-L, respectively. Consequently, MVP-cache also reduces the total energy consumption of the vector processor.

Specially, compared with the evaluation of the cache memory, the evaluation of the vector processor shows that the energy reduction by MVP-cache against MBC-S is decreased. This is because the portion of power reduction on tag arrays in the total processor is less than that in the cache memory. On the other hand, the energy reduction by MVP-cache against MBC-L in this evaluation is larger than that in the evaluation of cache memory. The reason is that, compared with MBC-L, the performance improvement by MVP-cache contributes to not only static energy reduction of the cache memory but also that of the total processor.

5. Conclusions

In order to match the demands of high data transfer performance and low energy consumption, MVP-cache has been designed and evaluated. It associates one tag array with multiple data arrays to reduce the energy consumption of tag arrays and improve the efficiency on short vector data transfers. Based on the performance evaluations with MMAs, the effects of MVP-cache are discussed in terms of data transfer performance improvement and energy reduction. The evaluation results show that MVP-cache can achieve comparable performance with the other competitive cache organizations, while the energy consumption of MVP-cache is smaller than those of the others. It is also found that the configuration of 16 data arrays associated with one tag array is a reasonable configuration for MMAs used in this paper.

As the future work of this paper, we will consider the implementation of MVP-cache by using 3D integration technologies to further reduce the energy consumption.

Acknowledgments

This research was partially supported by Core Research for Evolutional Science and Technology (CREST), Japan Science and Technology Agency (JST). This research was also partially supported by Grant-in-Aid for Scientific Research (B) No. 22300013, No. 25280041, and No. 25280012, and Grand-in-Aid for Exploratory Research No. 24650018, from the Ministry of Education, Culture, Sports, Science and Technology.

References

[1] International Telecommunication Union: ITU, “BT.2020: Parameter Values for Ultra-high Definition Television Systems for Production and International Programme Exchange,” http://www.itu.int/rec/R-REC-BT.2020-0-201208-I, 2012.

[2] M. Miura, K. Fudano, K. Ito, K. Aoki, H. Takizawa, and H. Kobayashi, “GPU implementation of phase-based stereo correspondence and its application,” 2012 IEEE International Conference on Image Processing, ICIP2012, pp.1697–1700, 2012.

[3] J.L. Hennessy and D.A. Patterson, Computer Architecture: A Quantitative Approach, Fifth Edition, Morgan Kaufmann, San Francisco, CA, USA, 2011.

[4] K. Diefendorff and P. Dubey, “How multimedia workloads will change processor design,” Computer, vol.30, pp.43–45, 1997.

[5] A. Musa, Y. Sato, T. Soga, K. Okabe, R. Egawa, H. Takizawa, and H. Kobayashi, "A shared cache for a chip multi vector processor," MEDEA '08: Proc. 9th workshop on MEmory performance, pp.24–29, 2008.
[6] D. Abts, A. Bataineh, S. Scott, G. Faanes, J. Schwarzmeier, E. Lundberg, T. Johnson, M. Bye, and G. Schwoerer, "The cray blackwidow: A highly scalable vector multiprocessor," SC '07: Proc. 2007 ACM/IEEE Conference on Supercomputing, pp.1–12, 2007.
[7] R. Espasa, F. Ardanaz, J. Emer, S. Felix, J. Gago, R. Gramunt, I. Hernandez, T. Juan, G. Lowney, M. Mattina, and A. Seznec, "Tarantula: A vector extension to the alpha architecture," Proc. 29th Annual International Symposium on Computer Architecture, pp.281–292, 2002.
[8] C. Batten, R. Krashinsky, S. Gerding, and K. Asanovic, "Cache refill/access decoupling for vector machines," MICRO 37: Proc. 37th Annual IEEE/ACM International Symposium on Microarchitecture, pp.331–342, 2004.
[9] A. Shahbahrami, B. Juurlink, and S. Vassiliadis, "A comparison between processor architectures for multimedia applications," Proc. 15th Annual Workshop on Circuits, Systems and Signal Processing, pp.138–152, 2004.
[10] C. Bienia, Benchmarking Modern Multiprocessors, Ph.D. thesis, Princeton University, Jan. 2011.
[11] A. Musa, Y. Sato, T. Soga, R. Egawa, H. Takizawa, K. Okabe, and H. Kobayashi, "Effects of MSHR and prefetch mechanisms on an on-chip cache of the vector architecture," International Symposium on Parallel and Distributed Processing with Applications, ISPA '08, pp.335–342, 2008.
[12] I. Kotera, K. Abe, R. Egawa, H. Takizawa, and H. Kobayashi, "Power-aware dynamic cache partitioning for CMPs," Trans. high-performance embedded architectures and compilers III, pp.135–153, 2011.
[13] R. Espasa, M. Valero, and J.E. Smith, "Out-of-order vector architectures," Proc. 30th Annual ACM/IEEE International Symposium on Microarchitecture, pp.160–170, 1997.
[14] A. Seznec and R. Espasa, "Conflict-free accesses to strided vectors on a banked cache," IEEE Trans. Comput., vol.54, no.7, pp.913–916, July 2005.
[15] Y. Gao, N. Shoji, R. Egawa, H. Takizawa, and H. Kobayashi, "Design and evaluation of a media-oriented vector processor with a multi-banked cache memory," 2013 11th IEEE Symposium on Embedded Systems for Real-Time Multimedia (ESTIMedia), pp.1–10, 2013.
[16] W. Dally and B. Towles, Principles and Practices of Interconnection Networks, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.
[17] K. Sewell, R. Dreslinski, T. Manville, S. Satpathy, N. Pinckney, G. Blake, M. Cieslak, R. Das, T. Wenisch, D. Sylvester, D. Blaauw, and T. Mudge, "Swizzle-switch networks for many-core systems," IEEE J. Emerging and Selected Topics in Circuits and Systems, vol.2, no.2, pp.278–294, 2012.
[18] T. Austin, E. Larson, and D. Ernst, "SimpleScalar: An infrastructure for computer system modeling," Computer, vol.35, no.2, pp.59–67, 2002.
[19] S. Li, J.H. Ahn, R.D. Strong, J.B. Brockman, D.M. Tullsen, and N.P. Jouppi, "McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures," Proc. 42nd Annual IEEE/ACM International Symposium on Microarchitecture, MICRO 42, pp.469–480, 2009.
[20] N. Muralimanohar, R. Balasubramonian, and N. Jouppi, "Optimizing nuca organizations and wiring alternatives for large caches with cacti 6.0," Proc. 40th Annual IEEE/ACM International Symposium on Microarchitecture, pp.3–14, 2007.
[21] C. Bienia, S. Kumar, J.P. Singh, and K. Li, "The PARSEC benchmark suite: Characterization and architectural implications," Tech. Rep., Princeton University, Jan. 2008.
[22] M. Li, R. Sasanka, S.V. Adve, Y. Chen, and E. Debes, "The ALPbench benchmark suite for complex multimedia applications," Proc. IEEE International Workload Characterization Symposium, pp.34–45, 2005.
[23] Y. Yokoya, Y. Kudoh, T. Hayasaka, J. Traeff, H. Ritzdorf, and Y. Hayashi, "The compilers and MPI library for SX-9," Tech. Rep., NEC Corporation, 2008.

Ye Gao received his B.E. degree in software engineering in 2007 from Dalian University of Technology, China. He received the M.S. and Ph.D. degrees of information sciences from Tohoku University in 2010 and 2014, respectively. He is currently a post-doctoral fellow in Cyberscience Center, Tohoku University. His research interests include computer architecture, parallel processing system, and multimedia processing technologies.

Masayuki Sato received the B.E. degree from Tohoku University in 2007. He also received the M.S. and Ph.D. degrees of information sciences from Tohoku University in 2009 and 2012, respectively. He is currently a post-doctoral fellow in Cyberscience Center, Tohoku University. His research interests include high-performance and low-power computer architecture.

Ryusuke Egawa received the B.E. degree, Master Degree on information sciences from Hirosaki University in 1999, 2001, respectively, and the Ph.D. degree on information sciences from Tohoku University in 2004. He is currently an associate professor of Cyberscience Center, Tohoku University. His research interests include computer architecture, VLSI design and high-performance computing.

Hiroyuki Takizawa is currently an associate professor of Graduate School of Information Sciences, Tohoku University. His research interests include high-performance computing systems and their applications. He received the B.E. Degree in Mechanical Engineering, and the M.S. and Ph.D. Degrees in Information Sciences from Tohoku University in 1995, 1997 and 1999, respectively. He is a member of the IEEE CS, and the IPSJ.

Hiroaki Kobayashi is currently Director and Professor of Cyberscience Center and Professor of the Graduate School of Information Sciences, Tohoku University. His research interests include high-performance computer architectures, grid and P2P computing, and multimedia applications. He received the B.E. Degree in Communication Engineering, and the M.E. and D.E. Degrees in Information Engineering from Tohoku University. He is a senior member of IEEE CS, and a member of ACM and IPSJ.