Memory Organization with Multi-Pattern Parallel Accesses

Arseni Vitkovski

ARCES – University of Bologna,
Viale Pepoli 3/2, 40123 Bologna, Italy
avitkovski@arces.unibo.it

Georgi Kuzmanov, Georgi Gaydadjiev

CE/EEMCS – Delft University of Technology,
Mekelweg 4, 2628 CD Delft, the Netherlands
{G.K.Kuzmanov, G.N.Gaydadjiev}@tudelft.nl

Abstract

We propose an interleaved memory organization supporting multi-pattern parallel accesses in two-dimensional (2D) addressing space. Our proposal targets computing systems with high memory bandwidth demands such as vector processors, multimedia accelerators, etc. We substantially extend prior research on interleaved memory organizations introducing 2D-strided accesses along with additional parameters, which define a large variety of 2D data patterns. The proposed scheme guarantees minimum memory latency and efficient bandwidth utilization for arbitrary configuration parameters of the data pattern. We provide mathematical descriptions and proofs of correctness for the proposed addressing schemes. The design complexity and the critical paths are evaluated using technology independent resource counts and confirm the scalability of the proposal. Hardware synthesis results for 90nm CMOS technology suggest that throughputs in the range between 44 and 1182 Gbit/s can be obtained at the cost of 26-212 Kgates for configurations of 2x2 32-bit up to 8x8 64-bit memory modules.

Index Terms—Conflict-free access, high bandwidth, multi-pattern access, parallel memories.

1. Introduction

The performance of modern SIMD machines and vector processors in particular, is highly dependent on the ability of the memory subsystem to rapidly feed the processing units with data from the main memory storage. Conventional cache memories proved to be reasonable solutions for increasing memory bandwidth in general purpose scalar systems, and significantly improve performance of applications with linear data locality. Such caches, however, fail to provide any performance benefits to vector machines or to applications with spatial locality such as multidimensional (e.g., 2D) matrix computations, multimedia, etc. In SIMD processors, this memory performance bottleneck is often prevented by interleaved memory systems, also referred to as space-multiplexed memories [1]. In these memory systems, parallel memory modules are organized to access application specific data patterns and to feed the processing units with data in a concurrent manner. A *module assignment function* and *a module row address function* (*skewing schemes*) ensure conflict-free parallel data access [2]. An important parameter, constituting the data access pattern is the *stride*, defined as the address distance between two adjacent data pattern elements in the original data alignment. As indicated in [3], no single skewing scheme exists allowing conflict-free accesses for all constant strides at any location in one dimensional interleaved memory organizations, let alone in two dimensional ones.

In this paper, we address the memory access problem of multi-pattern data access in two-dimensional interleaved memories. Our approach is to divide the problem into six trivial sub-problems. We consider an exhaustive set of pattern definition parameters and propose a performance efficient, interleaved memory organization. More specifically, the main contributions of the current proposal are as follows:

- Extended set of 2D pattern access parameters – base address; vertical and horizontal strides, group lengths, and block sizes;
- Support for all possible 2D data patterns described by the above parameters;
- Run-time programmability of the memory access pattern;
- Independency of the data pattern size from the number of the interleaved memory modules;
- Arbitrary-strided accesses for a minimal number of clock cycles;
- Modular implementation which can be easily simplified for a restricted subset of 2D data patterns;
- High design scalability confirmed by hardware synthesis results.

Theoretical estimations of the design complexity suggest that it is proportional to the square root of the number of memory modules. This is confirmed by synthesis results for 90nm CMOS technology, which indicate that a 32-bit 2×2 organization requires 26 Kgates and a 64-bit 8×8 one – 212 Kgates. The critical path is proportional to the logarithm of the number of memory modules, confirmed by the synthesized operating frequencies, which vary from 377 MHz for 2x2 32-bit modules down to 310 MHz for 8x8 64-bit modules. The maximum throughput,

978-3-9810801-3-1/DATE08 © 2008 EDAA

attainable by the synthesized 8x8 64-bit design, is 1182 Gbit/s.

The reminder of the paper is organized as follows: in Section 2, some related work and necessary theoretical background is provided. The proposed memory access scheme is described in Section 3. Its hardware implementation and complexity evaluation are presented in Section 4. The experimental results are reported in Section 5. Finally, Section 6 concludes the paper.

2. Related work and theoretical basis

Our goal is to develop a memory hierarchy with dynamically adjustable regular 2D access patterns, which would improve the data throughput between the main memory and processing units. We target highly data-parallel applications with regular data patterns, such as audio/video compression (ADPCM, G721, GSM, MPEG4, JPEG) or scientific vector calculations.

Related work: A number of solutions for conflict-free parallel memory access have been proposed in the literature. Vector processor designers have developed memory systems that are capable to deliver data at the required bandwidth to high pipeline numbers, e.g. [4], [7]. Various solutions have been proposed for optimal alignment of data in multiple parallel memory models [3], [5], [6], [7], [8]. Module assignment and row address functions have been used in different interleaved memories to improve their performance. In graphical visualization systems, researches have been investigating various data patterns, such as rectangles, horizontal and vertical lines, forward and backward diagonals [5], [9]. Other researches explore memory scheduling of DRAM chips by addressing locality characteristics within the 3D (bank, row, column) memory structure [10].

The above mentioned works lack architectural flexibility as the data access pattern is defined once at design time and cannot be changed at a later stage. Moreover, in previous work a number of restrictions on the access pattern parameters are applied [3], [5], [6], narrowing the application domain and increasing the programming efforts required. Our design is essentially an extension and generalization of the solution proposed in [6] preserving all its design advantages. Contrary to related work, our proposal provides a complete analysis of the pattern access parameters. As a result, we suggest design problem partitioning into six subtasks that cover arbitrary combinations and dimensions of the considered access pattern parameters.

Data pattern parameters: In interleaved memory organizations, the data distribution among the parallel memory modules is determined by a *module assignment function m*. A data element with a linear address *a* is assigned to a memory module according to *m(a)*. A row address function *A* determines the physical address of data element *a* inside a memory module. Fig. 1 illustrates the data pattern parameters we consider, namely: $W \in \mathbb{N}$ - data word width in bytes; $w_A \in \mathbb{N}$ - row address width in bits; $b'(vb,hb) \in [0,\mathbb{N}]$ - linear base address of the accessed block with 2D constituents *vb* and *hb*; $VS,HS \in \mathbb{N}$ - vertical and horizontal strides (in words); $VGL,HGL \in \mathbb{N}$ - respective group lengths (in words); $VBL,HBL \in \mathbb{N}$ - respective block lengths (in groups); $M \times N \in \mathbb{N}$ - the memory dimensions; $VD \times HD \in \mathbb{N}$ - size of the matrix of memory modules; (*i,j*) - vertical/horizontal group indices; (*k,l*) - respective indices of the elements inside a group. Fig. 1 depicts an example of a data pattern of six groups of size $VGL \times HGL = 2 \times 4$ and strides $(VS,HS) = (4,5)$.

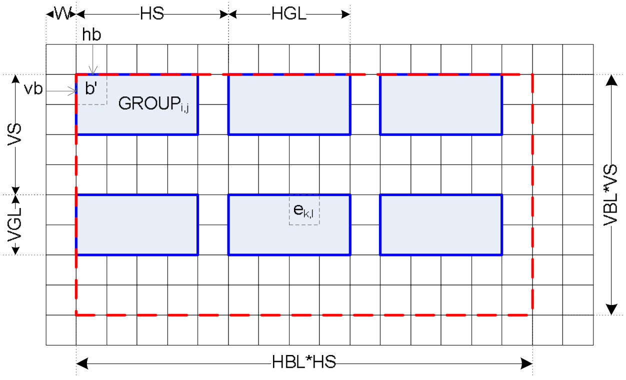

Fig. 1. Considered memory access pattern.

Theoretical basis: We first need to translate the linear base address b' into a 2D-address (vb,hb):

$$b' = vb \cdot N + hb,\ vb = \lfloor b'/N \rfloor,\ hb = b' \bmod N \quad \textbf{(1)}$$

Furthermore, any data element inside an accessed data block has the following 2D address (*va,ha*):

$$a' = va \cdot N + ha,$$
$$va = vb + i \cdot VS + k = a_{i,k},\ ha = hb + j \cdot HS + l = a_{j,l} \quad \textbf{(2)}$$

where $i \in [0, VBL-1]$, $k \in [0, VGL-1]$; $j \in [0, HBL-1]$, $l \in [0, HGL-1]$. Equation (2) suggests that the 2D address is completely separable, i.e. its vertical and horizontal constituents are not correlated. Moreover, they are identical. Therefore, in the following discussion we shall skip the direction prefixes 'V' and 'H' in the parameters notations.

The *stride* parameter *S*, can be represented as [3]:

$$S = \sigma \cdot 2^s \in \mathbb{N}, \quad \textbf{(3)}$$

where $\sigma = 2 \cdot k + 1$, $\forall k \in \mathbb{N}$ and $s \in [0,\mathbb{N}]$. Consequently, *S* is *odd* when *s*=0, and *even* for $s \in \mathbb{N}$. Consider the following theorems.

Theorem 1: No single skewing scheme can be found that allows conflict-free access for all the constant strides and group lengths when the data pattern can be unrestrictedly placed. The theorem is valid for arbitrary number of memory modules, when at least two data elements are accessed concurrently[1].

Proof: Let *m(a)* be the module assignment function and $a_{id} = B + j \cdot S + l$ the first accessed data element. Then, the next accessed data element is:

$$\begin{cases} a_{id+1} = B + j \cdot S + l + 1 = a_{id} + 1, & if\ l < GL - 1; \\ a_{id+1} = B + (j+1) \cdot S = a_{id} + S - l, & if\ l = GL - 1. \end{cases}$$

The two elements can not be accessed conflict-free if there are stride *S* and group length *GL* such that *m(aid)=m(aid+1)* □

Theorem 2: All *odd strides* can be accessed conflict-free with the low-order interleaved scheme if the number of memory modules is a power of two: *D=2d*, *d* ∈ℕ [11].

Theorem 3: Let stride *S′=2s*, group length *GL=2gl* , and number of memory modules along one dimension *D=2d* , where *s,gl,d*∈ℕ. Also let $s \geq d$ which means that the strides have less than one access per row. Then:

$$m(a) = \left(a + GL \cdot \left\lfloor \frac{\lfloor a/D \rfloor}{2^{s-d}} \right\rfloor \right) \bmod D \qquad \textbf{(4)}$$

allows conflict-free accesses to *D* memory modules.

Proof: We find a period *P* of function (4). If function *m(a)* maps address *a* to its module address for a stride *S′* and group length *GL*, then *m(a)=m(a+P·S′)*, ∀ *a* . Let us transform (4) as follows:

$$m(a) = \left((a + P \cdot S') + GL \cdot \left\lfloor \frac{\lfloor (a + P \cdot S')/D \rfloor}{2^{s-d}} \right\rfloor \right) \bmod D =$$

$$= \left(a + GL \cdot \left\lfloor \frac{\lfloor a/D \rfloor}{2^{s-d}} \right\rfloor + P \cdot (S' + GL) \right) \bmod D$$

According to the properties of the modulo operation, this equality holds when (*P*·(*S′*+*GL*)) mod *D*=0, which corresponds to the following set of periods:

$$\begin{bmatrix} P_1 \cdot (S' + GL) = x \cdot D; \\ P_2 \cdot S' = y \cdot D; \\ P_3 \cdot GL = z \cdot D, \end{bmatrix} \Rightarrow \begin{bmatrix} P_1 = x \cdot D/(S' + GL); \\ P_2 = y \cdot D/S'; \\ P_3 = z \cdot D/GL, \end{bmatrix}$$

where *x,y,z*∈ℕ and *x,y,z*:*P*∈ℕ. Thus, the minimum period is: $P_{\min} = D/GCD(D,GL)$, where GCD is greatest common divisor.

Now, we find number of accesses performed to the distinct memory modules. Harper and Jump showed in [12] that for a basic skewing storage scheme the number of distinct modules referenced during a vector access is $A' = \min(P, D)$. For our case *GL=2gl*, *gl*∈ℕ and the number of distinct modules is:

$$A = \min(P \cdot GL, D) = \min\left(\frac{D \cdot GL}{GCD(D, GL)}, D \right) = \min(LCM(D, GL), D) = D.$$

Thus, any vector of length *D*, having the form of Fig. 1 and (2), inside the sequence of module addresses generated by *m(a)* has exactly *D* distinct addresses. This is the definition of a conflict-free access. □

Theorem 4: Let stride *S′=2s*, group length *GL=2gl*, and number of memory modules along one dimension *D=2d*, where *s,gl,d* ∈ℕ. Also let *s<d* which means that the strides have at least one access per row. Then

$$m(a) = (a + \left(GL \cdot \left\lfloor \frac{a}{D} \right\rfloor \right) \bmod S') \bmod D \qquad \textbf{(5)}$$

allows conflict-free parallel accesses to *D* modules.

Proof: Identically to Theorem 3 we find a period *P* based on $m(a) = m(a + P \cdot S')$, ∀*a*.

$$m(a) = (a + P \cdot S + \left(GL \cdot \left\lfloor \frac{a + P \cdot S'}{D} \right\rfloor \right) \bmod S') \bmod D.$$

Using properties of the modulo operation, we derive:

$$\begin{cases} (P \cdot S') \bmod D = 0; \\ (a + \left(GL \cdot \left\lfloor \frac{a}{D} \right\rfloor \right) \bmod S') \bmod D = \\ = (a + \left(GL \cdot \left\lfloor \frac{a}{D} + \frac{P \cdot S'}{D} \right\rfloor \right) \bmod S') \bmod D. \end{cases}$$

The first equation gives the set of periods $P = x \cdot D/S'$, *x*∈ℕ: *P*∈ℕ. Substitute *P* in the second equation:

$$m(a) = (a + \left(GL \cdot \left\lfloor \frac{a}{D} + x \cdot \frac{D}{S'} \cdot \frac{S'}{D} \right\rfloor \right) \bmod S') \bmod D =$$

$$= (a + \left(GL \cdot \left\lfloor \frac{a}{D} \right\rfloor + GL \cdot x \right) \bmod S') \bmod D.$$

Using the same property of modulo operator we obtain: (*x*·*GL*) mod *S′*=0 ⇒ $x = y \cdot S'/GL$, where *y*∈ℕ: *x*∈ℕ. Substituting x provides the set of periods P :

$$P = x \cdot \frac{D}{S'} = y \cdot \frac{S'}{GL} \cdot \frac{D}{S'} = y \cdot \frac{D}{GL}, y \in \mathbb{N}: P \in \mathbb{N}.$$

This set of periods is identical to the one from the proof of Theorem 3. Therefore, the minimum period equals to $P_{\min} = D/GCD(D,GL)$ and, as follows from the same proof, the number of distinct addresses equals to *D*. Thus, the module assignment function (5) is conflict-free. □

Theorem 5: If a vector is to be accessed with even stride *S*=*σ* · 2*s*, where *GCD* (*σ*,2)=1, *s* ∈ℕ, *s* ≠ 0, and its elements are arranged in memory according to the storage scheme for a stride *S′=2s*, the accesses are conflict-free.

Proof: The proof is similar to [3] with the only difference that, we examine the sequence of groups of module addresses instead of the sequence of single addresses. □

3. Proposed memory access scheme

Since no single scheme exists for all strides and group lengths (Theorem 1), we propose to partition the problem in a number of cases. This reduces the problem to a set of trivial sub-problems (see Fig. 2). The initial problem partitioning is done based on the *stride oddness*. Theorem 2 suggests that odd strides can be accessed conflict-free

[1] This theorem follows Theorem 1 from [1] but in our case it is generalized since it considers the group length together with the stride.

using a basic skewing scheme [1], [6]. Furthermore, all odd and even strides are divided into six subgroups.

3.1 Module assignment function

We partition the design problem, imposed by the multiplicity of access patterns, into trivial subtasks. A module assignment function is devised for each of six different cases with respect to particular initial conditions:

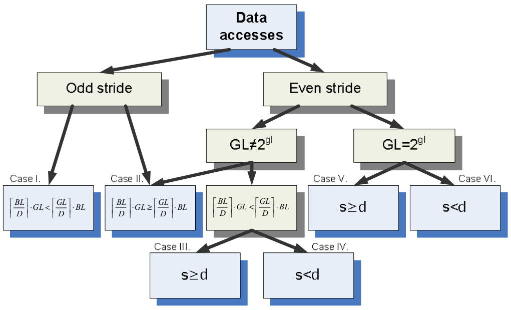
Data accesses
Odd stride
Even stride
GL≠2^gl
GL=2^gl
Case I.
$\left\lceil \frac{BL}{D} \right\rceil \cdot GL < \left\lceil \frac{GL}{D} \right\rceil \cdot BL$
Case II.
$\left\lceil \frac{BL}{D} \right\rceil \cdot GL \geq \left\lceil \frac{GL}{D} \right\rceil \cdot BL$
$\left\lceil \frac{BL}{D} \right\rceil \cdot GL < \left\lceil \frac{GL}{D} \right\rceil \cdot BL$
Case V.
s≥d
Case VI.
s<d
Case III.
s≥d
Case IV.
s<d

Fig. 2. Problem partition.

Case I. Initial conditions:

$$S = \{\sigma \cdot 2^s \mid GCD(\sigma,2) = 1; s = 0\}, \left\lceil \frac{BL}{D} \right\rceil \cdot GL < \left\lceil \frac{GL}{D} \right\rceil \cdot BL. \qquad \textbf{(6)}$$

The inequality guarantees that the number of accesses required for the complete pattern is minimal. When *BL>D,* more than one access are required to reed/write a whole group of data. If *D–*(*BL mod D*)>0, then the remaining memory modules stay unused. This case represents a conventional interleaved scheme with stride access which can be implemented conflict-free according to Theorem 2. Thus, the module assignment function is:

$$m(a) = a \bmod D, \qquad \textbf{(7)}$$

The indices iterate according to the following sequence:

$$\begin{aligned}(\mathrm{i,k}) = \ & ((0,0);(1,0);\ldots;(\mathrm{VBL}\text{-}1,0);\\ & (0,1);(1,1);\ldots;(\mathrm{VBL}\text{-}1,1);\\ & \ldots;\\ & (0,\mathrm{VGL}\text{-}1);(1,\mathrm{VGL}\text{-}1);\ldots;(\mathrm{VBL}\text{-}1,\mathrm{VGL}\text{-}1)).\end{aligned} \qquad \textbf{(8)}$$

The number of read/write accesses for the complete data pattern is equal to:

$$t = \left\lceil \frac{BL}{D} \right\rceil \cdot GL \cdot \qquad \textbf{(9)}$$

Case II. Initial conditions:

$$\begin{gathered} S = \{\sigma \cdot 2^s \mid GCD(\sigma,2) = 1; s = 0\}, \\ S = \{\sigma \cdot 2^s \mid GCD(\sigma,2) = 1; s \neq 0\} \ \& \ GL \neq 2^{gl}, \\ \left\lceil \frac{BL}{D} \right\rceil \cdot GL \geq \left\lceil \frac{GL}{D} \right\rceil \cdot BL \cdot \end{gathered} \qquad \textbf{(10)}$$

An access to the data pattern is performed group-wise. When *GL>D,* multiple accesses are required to reed/write a complete group. If *D*-(*GL* mod *D*)>0, there are unused memory modules. The inequality again guarantees minimal number of accesses cycles. It has been shown in [6] that any separate group inside the block can be accessed conflict-free.

The module assignment function is the same as for case I (see (7)), but the sequence of indices is different:

$$\begin{aligned}(i,k) = \ & ((0,0);(0,1);\ldots;(0,VGL\text{-}1);\\ & (1,0);(1,1);\ldots;(1,VGL\text{-}1);\\ & \ldots;\\ & (VBL\text{-}1,0);(VBL\text{-}1,1);\ldots;(VBL\text{-}1,VGL\text{-}1)).\end{aligned} \qquad \textbf{(11)}$$

The number of the read/write accesses is:

$$t = \left\lceil \frac{GL}{D} \right\rceil \cdot BL \cdot \qquad \textbf{(12)}$$

Case III. Initial conditions:

$$\begin{gathered} S = \{\sigma \cdot 2^s \mid GCD(\sigma,2) = 1; s \neq 0\} \ \& \ GL \neq 2^{gl}, \\ \left\lceil \frac{BL}{D} \right\rceil \cdot GL < \left\lceil \frac{GL}{D} \right\rceil \cdot BL, \\ s \geq d. \end{gathered} \qquad \textbf{(13)}$$

In this case, the indices iterate as in (8), but we employ the module assignment function proposed in [3]:

$$m(a) = \left(a + \left\lfloor \frac{\lfloor a/D \rfloor}{2^{s-d}} \right\rfloor \right) \bmod D \cdot \qquad \textbf{(14)}$$

In fact, all initial conditions for this case exactly repeat the ones used in [3]. The number of read/write accesses to the complete data pattern is described by (9).

Case IV. Initial conditions:

$$\begin{gathered} S = \{\sigma \cdot 2^s \mid GCD(\sigma,2) = 1; s \neq 0\} \ \& \ GL \neq 2^{gl}, \\ \left\lceil \frac{BL}{D} \right\rceil \cdot GL < \left\lceil \frac{GL}{D} \right\rceil \cdot BL, \quad s < d. \end{gathered} \qquad \textbf{(15)}$$

Again, the memory access repeats the sequence (8), and the module assignment function is borrowed from [3]:

$$m(a) = \left(a + \left\lfloor \frac{a}{D} \right\rfloor \bmod S' \right) \bmod D \cdot \qquad \textbf{(16)}$$

The amount of read/write accesses to the complete data pattern is described by (9).

Case V. Initial conditions:

$$S = \{\sigma \cdot 2^s \mid GCD(\sigma,2) = 1; s \neq 0\} \ \& \ GL = 2^{gl}, \ s \geq d. \qquad \textbf{(17)}$$

The data pattern access is element-wise, that allows the complete utilization of the memory modules. The module assignment function has the following representation:

$$m(a) = \left(a + GL \cdot \left\lfloor \frac{\lfloor a/D \rfloor}{2^{s-d}} \right\rfloor \right) \bmod D, \qquad \textbf{(18)}$$

The sequence of indices *(i, k)* is not important in this case since all memory modules are accessed conflict-free (Theorem 3). The amount of read/write accesses to the complete data pattern is:

$$t = \left\lceil \frac{GL \cdot BL}{D} \right\rceil. \qquad \textbf{(19)}$$

Case VI. Initial conditions:

$$S = \{\sigma \cdot 2^s \mid GCD(\sigma,2) = 1; s \neq 0\} \ \& \ GL = 2^{gl}, \ s < d. \qquad \textbf{(20)}$$

An access to the data pattern is performed element-wise, identical to case III. The module assignment function has the following representation:

$$m(a) = (a + \left(GL \cdot \left\lfloor \frac{a}{D} \right\rfloor \right) \bmod S') \bmod D, \qquad \textbf{(21)}$$

The sequence of indices *(i, k)* is not important again since all memory modules are accessed conflict-free (Theorem 4). The number of the read/write accesses to the complete data pattern is described by (19).

3.2 Row address function and access latencies

The row address function determines the linear address inside a memory module. In spite of having six different representations of the module assignment function, there is only one row address function. The row address function is separable and it is described as follows:

$$A(va,ha)=\left\lfloor \frac{va}{VD} \right\rfloor \cdot \left(\frac{N}{HD}\right)+\left\lfloor \frac{ha}{HD} \right\rfloor. \tag{22}$$

The total number of read/write accesses is described by the following best- and the worst case equations:

$$t_m^{best}=\left\lceil \frac{VGL\cdot VBL}{VD} \right\rceil \times \left\lceil \frac{HGL\cdot HBL}{HD} \right\rceil, \tag{23}$$

$$t_m^{worst}=\left(\min(VGL,VBL)\cdot\left\lceil \frac{\max(VGL,VBL)}{VD} \right\rceil\right)\times \\ \times\left(\min(HGL,HBL)\cdot\left\lceil \frac{\max(HGL,HBL)}{HD} \right\rceil\right). \tag{24}$$

The best case memory latency corresponds to the cases V-VI, while the worst case memory latency corresponds to the cases I-IV. Our primary design goal in all six cases, described in section 3.1, is to provide minimum possible latencies for the particular initial conditions.

4. Design implementation and complexity evaluation

In order to evaluate our scheme, we have verified it using a MatLab model, implemented it in VHDL and performed technology independent complexity evaluation. A completed RTL design was synthesized using Synopsys tools. Structurally, the interleaved memory consists of an address generation part, a data routing part, and a matrix of memory modules (see Fig. 3). The pattern parameters are stored in programmable Special Purpose Registers (SPRs). The address part is split in identical vertical and horizontal sides and includes the following blocks: mode select, address generator, set of row address generators, module assignment and address shuffle. The data routing part consists of a number of de-shuffle units. The critical path (highlighted on Fig. 3) comprises the address generator, the module assignment, and the decoding part of the shuffle block.

Mode select unit. This unit sets the address generation logic to a mode, corresponding to the six cases of the problem partitioning (section 3.1). The pattern parameters *S*, *GL* and *BL* are loaded from the SPRs. The hardware complexity is constant.

Address generator. The address generator produces the vertical and horizontal constituents of the 2D addresses according to (2). Data pattern parameters are loaded from the SPRs and an address mode is loaded from the mode select unit. The address generator consists of two double parallel counters (cases I-IV) and one simple parallel counter (cases V-VI), that generate sequences of paired indices *(i, k)* or *(j, l)* (Section 3.1).The complexity of this block is *O(wA·D)*. The critical path comprises a double counter, a multiplexer, a multiplier and an adder. The critical path complexity is *O(log D)*.

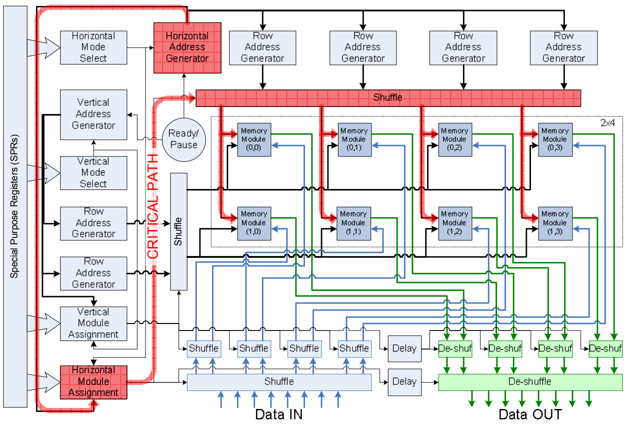

Fig. 3. Parallel memory controller block diagram.

Row address generator. This unit translates vertical/horizontal constituents of 2D addresses into physical linear addresses inside the memory modules (called row address) according to (22). Since (22) is separable, vertical blocks are connected to upper address bits, and horizontal blocks - to lower bits. No additional logic is needed to implement this block.

Module assignment unit. This unit transforms vertical/horizontal constituents of 2 addresses into memory module addresses inside the memory modules matrix according to (7), (14), (16), (18), and (21). Data pattern parameters are read from SPRs and an address mode is loaded from the mode select block. The equations are implemented in parallel and their outputs are multiplexed according to the address mode as depicted on Fig. 2. The complexity is *O(log D)*. The critical path comprises a masking unit, two shifters, an adder, and a multiplexer. It is mostly influenced by the adder of width *log D* and it is *O(log(log D))*.

Shuffle unit. The shuffle unit is used to reorder data according to the module assignment function. It consists of a parallel set of de-multiplexers and output OR-gates. The complexity of the shuffle unit is *O(wA·D)* and the critical path does not depend on D or *wA*.

De-shuffle unit. The de-shuffle unit reorders data back to the initial sequence. It consists of a set of multiplexers and its complexity is *O(wA·D)*.

5. Synthesis results and analysis

RTL simulation proved the functional correctness of the implemented design. The length of the data stream depends on the input parameters and is computed using equations (9), (12) or (19). Its width depends on the size of the matrix of memory modules *VD×HD* and data word width *W*.

Our technology independent complexity estimation indicated that the logic complexity is *O(wA·D)* (while the complexity of the matrix of memory modules is O(D^2)) since all address transformations are implemented per vertical and horizontal constituencies rather than per separate memory module. The critical path complexity is *O(log D)*, i.e. it is weakly sensitive to the size of the memory matrix. In fact, the throughput is directly proportional to the matrix size *VD×HD*, and inversely – to the critical path, i.e. $throughput \propto W \cdot D^2 / \log D$.

Table I. Synthesis results with 10-bit row address width for 90nm CMOS technology.

Matrix size	Complexity (KGates)		Frequency (MHz)		Throughput (Gbits/sec)	
	W=4	*W*=8	*W*=4	*W*=8	*W*=4	*W*=8
2×2	25.34	26.73	377	371	44.94	88.45
2×4	33.81	39.11	341	336	81.30	160.21
2×8	58.48	70.19	314	321	149.72	306.12
4×4	46.60	53.27	336	333	160.21	317.57
4×8	88.07	101.57	321	314	306.12	598.90
8×8	176.83	211.06	313	310	597.00	1182.55

The synthesis was performed for an ASIC 90nm CMOS technology. The results for six different matrix sizes, word widths *W* of 4 and 8 Bytes, and 10-bit addresses are presented in Table I. In fact, data word width of 8 Bytes corresponds to utilization of two concurrently coupled 32-bit wide memory modules. The presented synthesis results confirm the linear increase of the design complexity and the quadratic increase of the throughput, derived from our theoretical estimations. As it was expected, the critical path is proportional to the logarithm of the matrix size and the design complexity linearly depends on the matrix size. This fact gives an advantage of scaling the design according to a particular problem's requirements with some minor degradation of speed. More extensive evaluation discussions on the proposed design and results from its physical implementation are presented in [12].

6. Conclusions

We presented an interleaved memory organization for highly parallel computing systems such as vector processors and media accelerators. A major contribution of this work was the proposed partitioning of the memory access problem into six subtasks. This approach allowed a complete design solution, which supports complex data access patterns for interleaved memory systems. More specifically, compared to related art, our design proposal introduced the following new features: support of an extended set of 2D access pattern parameters; programmability of the access patterns via SPRs; arbitrary dimensions of the access pattern; design scalability. We proved theoretically that the design complexity scales proportionally to *O(wA·D)* and the critical path is proportional to *O(log D)*. Our theoretical conclusions were confirmed by mathematical modeling and by actual hardware synthesis for 90nm CMOS technology. In the future, we plan to integrate the design into a reconfigurable platform and perform experiments on real applications.

Acknowledgements

This research was partially supported by the Dutch Technology Foundation STW, applied science division of NWO and the Technology Program of the Dutch Ministry of Economic Affairs (project DCS.7533); European Commission FP6 (projects SARC and MORPHEUS); and European Doctorate in Information Technology (EDITH).

References

[1] E. Aho, J. Vanne, and T.D. Hamalainen, "Parallel Memory Architecture for Arbitrary Stride Accesses," Design and Diagnostics of Elect. Circ. and Syst., pp. 63 – 68, April 18-21, 2006.

[2] J. Takala and T. Jarvinen, "Stride permutation access in interleaved memory systems," in Domain-Specific Multiproc. – Syst., Architect., Modeling, and Simul., pp. 63-84, Mercel Dekker, New York, NY, USA, 2004.

[3] D.T. Harper III, D.A. Linebarger, "Conflict-Free Vector Access Using a Dynamic Storage Scheme," IEEE Trans. on Comp., vol. 40, no. 3, pp. 276-283, March 1991.

[4] P. Budnik and P.J. Kuck, "The organization and use of parallel memories," IEEE Trans. on Comp., vol. 20, no. 12, pp. 1566-1569, 1971.

[5] J.W. Park, "An efficient buffer memory system for subarray access," IEEE Trans. on Parallel and Distrib. Syst., vol. 12, no. 3, pp. 316-335, 2001.

[6] G. Kuzmanov, G. Gaydadjiev, and S. Vassiliadis, "Multimedia rectangular addressable memory," IEEE Trans. on Multimedia, vol. 8, no. 2, pp. 315-322, 2006.

[7] D.H. Lawrie, "Access and alignment of data in an array processor," IEEE Trans. on Comput., vol. C-24, no. 12, pp. 1145-1155, 1975.

[8] A. Seznec and R. Espasa, "Conflict free accesses to strided vectors on a banked cache," IEEE Trans. on Comp., vol. 54, pp. 913 - 916, 2005.

[9] J. Lee, C. Park, and S. Ha, "Memory access pattern analysis and stream cache design for multimedia applications," Proc. of the ASP-DAC 2003 – Design Automation Conf., pp. 22-27, 2003.

[10] S. Rixner, W.J. Dally, U.J. Kapasi, P. Mattson, and J.D. Owens, "Memory access scheduling," Proc. of the 27th Int. Symp. on Comp. Arch., pp. 128-138, 2000.

[11] M. Valero, T. Lang, M. Peiron, E. Ayguade, "Conflict-free access for streams in multimodule memories," IEEE Trans on Comp., vol. 44, no. 5, pp. 634-646, 1995.

[12] A. Vitkovski, "Architecture and implementation of the 2D memory with multi-pattern parallel accesses", tech. rep., no. CE-TR-2007-03, CE/EEMCS, TUDelft.