A novel parallel memory organization supporting multiple access types with matched memory modules

Sheng Liu, Shuming Chen, Hu Chen, and Yang Guo

Computer School, National University of Defense Technology
#109, Deya Road, Changsha, 410073, China
a) liusheng83@gmail.com

Abstract: This paper introduces a Bilinear Skewed Parallel Memory (BilisPM), which can support multiple conflict-free access types and the circular addressing in X-Y directions of the 2D space. BilisPM features matched Memory Modules (MMs) and can effectively save the on-chip area. We introduce the formal specifications of BilisPM and give its hardware implementation. Experimental results show that BilisPM can reduce the chip area by 22.7% on average (38.1% at most), and its controller consumes smaller chip area at reasonable critical path delay, as compared with the traditional schemes with unmatched MMs.

Keywords: parallel memory, multiple access types, mapping functions

Classification: Integrated circuits

References


1 Introduction

The Parallel Memory (PM) organizations, which consist of several Memory Modules (MMs), have been widely used in SIMD processors and are especially suitable for the video/image algorithms requiring 2D space accesses.

Conventional simple PM schemes [1, 2, 3] cannot simultaneously support enough access types without conflicts due to their matched MMs. Here “matched” means that if the SIMD data path width is M×N, the number of needed MMs will be also M×N. Park [4, 5] uses P MMs (P is a prime number larger than M×N) to support accesses by row, column, block and diagonal. However, this approach introduces redundant MMs and cannot provide the circular addressing. C. Liu [6] fixes the limitations of Park’s scheme using 2M×N MMs which results in more chip area cost.

We propose a novel Bilinear Skewed Parallel Memory (BilisPM). It features matched MMs and the width of each MM is extended from one byte to two bytes. By proposed mapping functions, BilisPM can support conflict-free accesses by row, column, block and etc. BilisPM will not introduce extra chip area because it only changes the shape of each MM and needs no additional MMs. Besides, it uses the linear skewed method in the X and Y directions, so BilisPM can provide the circular addressing in the two 2D directions.

2 Theories of BilisPM

Our BilisPM has matched MMs, and the width of each MM is different from that of traditional schemes. So, it possesses the particular mapping functions. We first give the mapping functions of BilisPM and prove that BilisPM can support multiple conflict-free access types. Then, we formulate the definition and feature of the bank status to simplify the control logic of BilisPM.

2.1 Mapping functions

\[ s(x, y) = ((M + 1) \times (x/2) + y)\%(M \times N) \]  
\[ a(x, y) = x/2 + y/(M \times N) \times (X_m/2) \]  
\[ f(x, y) = x \% 2 \]

Here “/” and “\%” are the integer division and the modulo operations. Usually, M, N, X_m and Y_m are powers of two. \(f(x,y)=1\) (0) indicates the element \((x,y)\) is in the high (low) byte at the \(a(x,y)\)th position of MMs\([s(x,y)]\).

Fig. 1 shows the values of mapping functions when \(M \times N=4 \times 2\) and \(X_m=Y_m=16\). \{s(x,y),a(x,y)\} or \{s'(x,y),a(x,y)\} indicates that the element \((x,y)\) is mapped in the high or low byte at the \(a(x,y)\)th position of MMs\([s(x,y)]\).

2.2 Access types

BilisPM is aimed at simultaneously providing the conflict-free accesses by row, column and block, to greatly improve the video/image algorithms.

Definition 1 (Base Element, BE): The left/top/top-left element of an access by row, column or block is defined as the Base Element.

Definition 2 (Access Types): For a memory space of \(X_m \times Y_m\), Access
Fig. 1. Values of the mapping functions in BilisPM

**Types** with BE \((x_0,y_0)\) are defined in Eq. (4)~(7). Here ROW, COL, BLK and ROWS denote accesses by row, column, block and interval row.

\[
\text{ROW}(x_0, y_0) = \{(x_0 + u)\%X_m, y_0) | 0 \leq u \leq M \times N - 1\} \tag{4}
\]

\[
\text{COL}(x_0, y_0) = \{(x_0, (y_0 + v)\%Y_m) | 0 \leq v \leq M \times N - 1\} \tag{5}
\]

\[
\text{BLK}(x_0, y_0) = \{(x_0 + u)\%X_m, (y_0 + v)\%Y_m) | 0 \leq u \leq M - 1, 0 \leq v \leq N - 1\} \tag{6}
\]

\[
\text{ROWS}(x_0, y_0) = \{(x_0 + 2u)\%X_m, y_0) | 0 \leq u \leq M \times N - 1\} \tag{7}
\]

**Theorem 1**: If \(M \times N \geq 8\) (\(M = N\) or \(M = 2N\)), then BilisPM can support the access types described in Eq. (4)~(7) without conflicts.

**Proof.** Based on Theorem 2 (Section 2.3), we only need to prove that a certain access type with BE \((0,0)\) or \((1,0)\) is conflict-free. Since accesses by row, column and interval row are simple, we only prove the access by block. And we only prove the case when \(\text{BLK}(x_0,y_0) = \text{BLK}(0,0)\) due to the page limitation. By Eq. (1) and (6), the index numbers of MMs of \(\text{BLK}(0,0)\) are:

\[
((M+1) \times (u/2) + v)\% (M \times N), \quad 0 \leq u \leq M - 1, 0 \leq v \leq N - 1 \tag{8}
\]

When \(u = 0\), the results of Eq. (8) are 0, 1, ..., N-1.

......

When \(u = M-2\), the results of Eq. (8) are \((M+1) \times (u/2) + v)\% (M \times N), ((M+1) \times (u/2-1)) \% (M \times N), ..., ((M+1) \times (M/2-1)) \% (M \times N)\). If \(M \times N \geq 8\) (\(M = N\) or \(M = 2N\)), then 0 < \(u \leq M-1\) < \(N-1\) < \((M+1) \times (M/2-1)\) < ... < \((M+1) \times (M/2-1)+N-1\) < \(M \times N\). So, the results of Eq. (8) are different when 0 \(\leq u \leq M-1\), 0 \(\leq v \leq N-1\), and \(v \% 2 = 0\). And from Eq. (2) and (3), when \(v \% 2 = 1\), the elements \((u,v)\) and \((u-1,v)\) will be mapped into the high and low byte at the same position of a certain MM. So the block access is conflict-free. \(\square\)
2.3 Definition and feature of bank status

**Definition 3 (Bank Status, BS):** Let’s assume parameter \( k \) is an integer with two bits, then \( k[1]=1 \) or \( k[0]=1 \) denotes one element is mapped into the high or low byte at a position of MMs[T] in a certain access; The coordinate differences between the element of MMs[T] and BE are recorded as \( u \) and \( v \). When calculating the \( u \) and \( v \), we only use the element in the high byte if MMs[T] holds two elements. Then the vector \((k, u,v)\) is defined as Bank Status of MMs[T], and is recorded as \( T(k,u,v) \) whose components are noted as \( k_T, u_T \) and \( v_T \).

**Theorem 2:** \( \xi \in \{0,1\} \), and \( x_0\%2=\xi \). We suppose that the BS of MMs[T] in a certain access type with BE \((\xi,0)\) is \( T(k,u,v) \), and in the same access type with BE \((x_0,y_0)\), the element \((x_0+y_0,0)\) is mapped into MMs\([s_0]\) and the BS of MMs\([Q]\) is \( Q(k,u,v) \). If \( Q=(T+s_0)\%(M\times N) \), then \( Q(k,u,v)=T(k,u,v) \).

**Proof.** We prove the case when \( \xi=0 \), since the proof is similar when \( \xi=1 \).

If \( k_T=2^b10 \), to prove \( Q(k,u,v)=T(k,u,v) \), i.e., \( k_T=k_Q, u_T=u_Q \) and \( v_T=v_Q \), we need to prove: 1) the element \((x_0+u_T,y_0+v_T)\) is mapped into MMs\([Q]\); 2) the element \((x_0+u_T+1,y_0+v_T)\) isn’t mapped into MMs\([Q]\). We only prove 2) due to the page limitation. Obviously, MMs\([s_0]\) holds the element \((x_0,y_0)\) and MMs\([T]\) won’t hold the element \((u_T+1,v_T)\). From Eq. \((1)\), we can get

\[
((M+1)\times(x_0/2)+y_0)\%(M\times N) = s_0 \tag{9}
\]

\[
((M+1)\times((u_T+1)/2)+v_T)\%(M\times N) \neq T \tag{10}
\]

From Eq. \((9)\), we can get

\[
(M+1)\times(x_0/2)+y_0 = \alpha \times M \times N + s_0 \tag{11}
\]

Assuming the element \((x_0+u_T+1,y_0+v_T)\) is mapped into MMs\([Q]\), we can get \(((M+1)\times((x_0+u_T+1)/2)+y_0+v_T)\%(M\times N)=Q=(T+s_0)\%(M\times N)\), i.e.,

\[
((M+1)\times((x_0+u_T+1)/2)+y_0+v_T) = \beta \times M \times N + T + s_0 \tag{12}
\]

Eq. \((11)\) subtracted from Eq. \((12)\) gives \(((M+1)\times((x_0+u_T+1)/2-x_0/2)+v_T) = (\beta-\alpha)\times M \times N + T\). Considering \(x_0\%2=0\) and \(0\leq T<\!M\times N\), we can get

\[
((M+1)\times((u_T+1)/2)+v_T)\%(M\times N) = T \tag{13}
\]

Obviously, Eq. \((13)\) contradicts with Eq. \((10)\). So the assumption is invalid, i.e., the element \((x_0+u_T+1,y_0+v_T)\) is not mapped into MMs\([Q]\).

The proof of the case when \( k_T=2^b10 \) is over. Similarly, we can prove the cases when \( k_T=2^b01, k_T=2^b11 \) and \( k_T=2^b00 \). □

3 Implementation of BilisPM

From Theorem 2, we can design a BS Table (BST) to record the BS of each MM in different access types with BE \((0,0)\) or \((1,0)\). When dealing with an access with BE \((x_0,y_0)\), we only need to calculate the MM number \((s_0)\) of its BE and to read a certain item from the BST. Then we can calculate the initial addresses within MMs. After the initial addresses being circular right...
shifted by $s_0$ bits, we can obtain the real addresses within MMs. Besides, from Theorem 2, the positions of the data in an access also have the circular shifting relationship with the same access type with BE (0,0) or (1,0).

![Fig. 2. Hardware implementation of BilisPM](image)

Fig. 2 gives the hardware implementation of BilisPM and Table I shows the contents of the BST, when $M \times N = 4 \times 2$ and $X_m = Y_m = 16$. The controller of BilisPM mainly includes two parts: the data selection and the address generation. We will take a load request to BLK(3,4) as an example to show how the controller deals with the addresses and the data.

**Table I. Content of the BST**

<table>
<thead>
<tr>
<th>Types</th>
<th>$k_0$, $u_0,v_0$</th>
<th>$k_1$, $u_1,v_1$</th>
<th>$k_2$, $u_2,v_2$</th>
<th>$k_3$, $u_3,v_3$</th>
<th>$k_4$, $u_4,v_4$</th>
<th>$k_5$, $u_5,v_5$</th>
<th>$k_6$, $u_6,v_6$</th>
<th>$k_7$, $u_7,v_7$</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROW $x_0 % 2 = 0$</td>
<td>3.0, 0</td>
<td>3.6, 0</td>
<td>0, -</td>
<td>3.2, 0</td>
<td>0, -</td>
<td>3.4, 0</td>
<td>0, -</td>
<td></td>
</tr>
<tr>
<td>ROW $x_0 % 2 = 1$</td>
<td>1.0, 0</td>
<td>3.5, 0</td>
<td>0, -</td>
<td>3.1, 0</td>
<td>2.7, -</td>
<td>0, -</td>
<td>3.3, 0</td>
<td>0, -</td>
</tr>
<tr>
<td>COL $x_0 % 2 = 0$</td>
<td>2.0, 0</td>
<td>2.0, 1</td>
<td>2.0, 2</td>
<td>2.0, 3</td>
<td>2.0, 4</td>
<td>2.0, 5</td>
<td>2.0, 6</td>
<td>2.0, 7</td>
</tr>
<tr>
<td>COL $x_0 % 2 = 1$</td>
<td>1.0, 0</td>
<td>1.0, 1</td>
<td>1.0, 2</td>
<td>1.0, 3</td>
<td>1.0, 4</td>
<td>1.0, 5</td>
<td>1.0, 6</td>
<td>1.0, 7</td>
</tr>
<tr>
<td>BLK $x_0 % 2 = 0$</td>
<td>3.0, 0</td>
<td>3.0, 1</td>
<td>0, -</td>
<td>3.2, 0</td>
<td>3.2, 1</td>
<td>0, -</td>
<td>0, -</td>
<td>0, -</td>
</tr>
<tr>
<td>BLK $x_0 % 2 = 1$</td>
<td>1.0, 0</td>
<td>1.0, 1</td>
<td>0, -</td>
<td>3.1, 0</td>
<td>3.1, 1</td>
<td>0, -</td>
<td>2.3, 0</td>
<td>2.3, 1</td>
</tr>
<tr>
<td>ROWS $x_0 % 2 = 0$</td>
<td>2.0, 0</td>
<td>2.3, 0</td>
<td>2.6, 0</td>
<td>2.1, 0</td>
<td>2.4, 0</td>
<td>2.7, 0</td>
<td>2.2, 0</td>
<td>2.5, 0</td>
</tr>
<tr>
<td>ROWS $x_0 % 2 = 1$</td>
<td>1.0, 0</td>
<td>1.3, 0</td>
<td>1.6, 0</td>
<td>1.1, 0</td>
<td>1.4, 0</td>
<td>1.7, 0</td>
<td>1.2, 0</td>
<td>1.5, 0</td>
</tr>
</tbody>
</table>

According to the access type (block) and $x_0 \% 2 = 1$, the controller of BilisPM can read the BST and get $V_1 = \{\{1,0,0\}, \{1,0,1\}, \{0,-,\}, \{3,1,0\}, \{3,1,1\}, \{0,-,\}, \{2,3,0\}, \{2,3,1\}\}$. The initial addresses within MMs can be calculated as $\{1, -, 2, 2, -, 3, 3\}$. Then we can get $V_2 = \{\{1,1\}, \{1,1\}, \{0,-\}, \{3,2\}, \{3,2\}, \{0,-\}, \{2,3\}, \{2,3\}\}$. Meanwhile, the controller calculates the MM number of the element (3,4) and gets $s_0 = 7$. After being circular right shifted by $s_0$ bits, $V_2$ changes to $V_3 = \{\{1,1\}, \{0,-\}, \{3,2\}, \{3,2\}, \{0,-\}, \{2,3\}, \{2,3\}\}$, which are coherent with Fig. 1.
If the data from the MMs are \{a_1a_2, b_1b_2, c_1c_2, d_1d_2, e_1e_2, f_1f_2, g_1g_2, h_1h_2\}, they will change to \{h_1h_2, a_1a_2, b_1b_2, c_1c_2, d_1d_2, e_1e_2, f_1f_2, g_1g_2\} after being circular left shifted by \(s_0\) bits. According to the access type (block) and \(x_0\%2=1\), the final valid data are \{h_2, c_1, c_2, f_1, a_2, d_1, d_2, g_1\}.

4 Experimental results

4.1 Area of MMs

The area of MMs accounts for most of the area of PMs, so we first evaluate it. Table II compares the area of the MMs in BilisPM with that of baseline schemes [1, 2, 3], Park’s scheme [4, 5], and C. Liu’s scheme [6]. The results are obtained with Artisan Memory Compiler under SIMC 0.13 \(\mu\)m technology.

<table>
<thead>
<tr>
<th>Memory Capacity</th>
<th>Area (mm²)</th>
<th>BilisPM /Park</th>
<th>BilisPM /C. Liu</th>
</tr>
</thead>
<tbody>
<tr>
<td>16 KB</td>
<td>4×2</td>
<td>0.72</td>
<td>0.99</td>
</tr>
<tr>
<td></td>
<td>4×4</td>
<td>0.81</td>
<td>0.86</td>
</tr>
<tr>
<td></td>
<td>8×4</td>
<td>0.99</td>
<td>1.15</td>
</tr>
<tr>
<td></td>
<td>8×8</td>
<td>1.33</td>
<td>1.39</td>
</tr>
<tr>
<td>32 KB</td>
<td>4×2</td>
<td>1.10</td>
<td>1.51</td>
</tr>
<tr>
<td></td>
<td>4×4</td>
<td>1.44</td>
<td>1.53</td>
</tr>
<tr>
<td></td>
<td>8×4</td>
<td>1.61</td>
<td>1.86</td>
</tr>
<tr>
<td></td>
<td>8×8</td>
<td>1.98</td>
<td>2.08</td>
</tr>
<tr>
<td>64 KB</td>
<td>4×2</td>
<td>2.09</td>
<td>2.88</td>
</tr>
<tr>
<td></td>
<td>4×4</td>
<td>2.20</td>
<td>2.34</td>
</tr>
<tr>
<td></td>
<td>8×4</td>
<td>2.88</td>
<td>3.34</td>
</tr>
<tr>
<td></td>
<td>8×8</td>
<td>3.23</td>
<td>3.38</td>
</tr>
</tbody>
</table>

BilisPM has the similar area cost to that of baseline schemes, because BilisPM only changes the shape of each MM and does not introduce additional MMs. However, BilisPM can provide conflict-free accesses by row, column and block, which cannot be supported by baseline schemes.

Park’s scheme can also support accesses by row, column and block without conflicts. Compared with Park’s scheme, BilisPM can reduce the chip area by 17.1% on average (37.4% at most). That’s because Park’s scheme doesn’t address equivalent P-M×N MMs, which results in a waste of chip area. Besides, Park’s scheme can hardly provide the circular addressing, which will contribute to 35% of the external bandwidth [6].

BilisPM and C. Liu’s scheme have the similar functions: multiple access types and the circular addressing. However, BilisPM can reduce the chip area by 22.7% on average (38.1% at most), compared with C. Liu’s scheme which splits each MM into two smaller MMs and introduces extra area cost.

4.2 Hardware complexity of PM controllers

We use the estimating method similar as that in paper [2]: The area cost of
λ-input basic gates is assumed to be λ cost-units (CUs), with the exception that XOR/XNOR has 2λ CUs. For all the 2-input gates the delay is assumed to be 1 τ. We select $X_m = Y_m = 256$, $M \times N = 4 \times 4$, and $P = 17$.

The controller (3,578 CUs) of BilisPM consumes 87% area of the controller (4,123 CUs) of Park’s scheme, and 72% area of the controller (4,978 CUs) of C. Liu’s scheme. The reasons are: 1) The width of the shifter in BilisPM is smaller due to its matched MMs. 2) The number and width of the required table in BilisPM are also smaller due to its feature described in Theorem 2. 3) The number of adders of BilisPM is smaller than that of C. Liu’s scheme.

The critical path delay of BilisPM (21 τ) is shorter than that of C. Liu’s scheme (25 τ), and is a little longer than that of Park’s scheme (20 τ). The critical path delay of the three schemes are also composed of three parts: the table reading time ($T_{table}$), the address calculating time ($T_{addr}$) and the address shifting time ($T_{shift}$). Since it can read the BST directly by access types and the parity of $x_0$, BilisPM has a shorter $T_{table}$. The width of the barrel shifter in BilisPM is smaller, so BilisPM also has a shorter $T_{shift}$. Besides, due to the circular addressing roles, the $T_{addr}$ of BilisPM is equal to that of C. Liu’s scheme but is longer than that of Park’s scheme. Anyway, the critical path delay of BilisPM is reasonable.

### 5 Conclusion

This paper introduces BilisPM, which can support multiple conflict-free access types (row, column, block and etc.) and provide the circular addressing in X-Y directions of the 2D space. BilisPM can reduce the chip area by 22.7% on average (38.1% at most) and its controller has smaller chip area and reasonable critical path delay, as compared with schemes with unmatched MMs.

### Acknowledgments

This work is supported by NSF of China (61070036 and 61133007).