SCIENTIFIC PUBLICATIONS OF THE STATE UNIVERSITY OF NOVI PAZAR
SER. A: APPL. MATH. INFORM. AND MECH. vol. 2, 1 (2010), 11-20

# Matrix Multiplication on Linear Bidirectional Systolic Arrays

**E. I. Milovanović, B. M. Randjelović, I. Ž. Milovanović, M. K. Stojčev**

**Abstract:** This paper addresses the problem of rectangular matrix multiplication on bidirectional linear systolic arrays (SAs). We analyze all bidirectional linear SAs in term of number of processing elements, execution time and efficiency. We conclude that the efficiency depends on the relation between loop boundaries in systolic algorithm (i.e. matrix dimensions). We point out which SA is the best choice depending on the relation between matrix dimensions.

**Keywords:** matrix multiplication, linear systolic arrays, efficiency.

## 1 Introduction

Matrix multiplication plays a central role in numerical linear algebra, since one has to compute this product in several stages of almost all numerical algorithms, as well as in many technical problems, especially in the area of digital signal processing, pattern recognition, plasma physics, weather prediction, etc. Therefore, finding an efficient algorithm for performing these computations is at the focus of interest of many researchers. Matrix multiplication is a very regular computation and lends itself well to parallel implementation. Regular structures, such as systolic arrays (SAs), are well suited for matrix multiplication and are also amenable to VLSI implementation because of simple and regular design, and nearest-neighbor communications. A systolic system is a network of processing elements (PEs) that rhythmically compute and pass data through the system. Once a data item is brought from the memory, it can be used effectively in each PE as it passes while being "pumped" from cell to cell along the array.

Systolic arrays have been designed for a wide variety of computationally intensive problems in signal processing, numerical problems, pattern recognition, database and dictionary machines, graph algorithms etc. Systolic arrays implemented in silicon chips are typically laid out in a linear array or bidimensional grid of cells. One dimensional or linear systolic arrays are especially popular because of low number of I/O pins required for the interconnection with the "outside world".

To handle matrix multiplication, 2D and 1D systolic arrays have been proposed. Matrix multiplication on 2D arrays has been extensively studied. Most of the 1D arrays proposed

Manuscript received December 22, 2009 ; revised March 11, 2010; accepted May 20, 2010.

E. I. Milovanović, B. M. Randjelović, I. Ž. Milovanović, M. K. Stojčev are with the Faculy of Electronic Engineering, Niš, Serbia

for matrix multiplication are with one-dimensional links. However, these arrays require delay elements between successive processing elements, since input data have to pass through the array with different speed. This paper deals with multiplication of rectangular matrices on bidirectional linear systolic arrays (BLSA) with two-dimensional links. These arrays do not require delay elements for implementation of matrix multiplication.

Since matrix multiplication is a three-dimensional problem and BLSA is suitable for two-dimensional problems, the computation is performed by repeated computation of two-dimensional tasks: a) product of a matrix and column-vector, b) product of row-vector and matrix, or c) outer product of column-vector and row-vector. Each of the methods is explained and the corresponding systolic algorithms and BLSA that implement them are given in this paper. Then, we analyze obtained systolic arrays in term of efficiency.

## 2 Background

Let $A = (a_{ik})_{N_1 \times N_3}$ and $B = (b_{kj})_{N_3 \times N_2}$ be two rectangular matrices. Their product can obtained according to the following well-known recurrence equation

$$c_{ij}^{(k)} := c_{ij}^{(k-1)} + a_{ik} b_{kj}, \tag{1}$$

for $i = 1,2,\ldots,N_1$, $j = 1,2,\ldots,N_2$ and $k = 1,2,\ldots,N_3$. For systolic implementation of this product, the following algorithm can be used.

**Algorithm_1**
**for** $k := 1$ **to** $N_3$ **do**
**for** $j := 1$ **to** $N_2$ **do**
**for** $i := 1$ **to** $N_1$ **do**
$a(i,j,k) := a(i,j-1,k)$;
$b(i,j,k) := b(i-1,j,k)$;
$c(i,j,k) := c(i,j,k-1) + a(i,j,k) * b(i,j,k)$;

where $a(i,0,k) \equiv a_{ik}$, $b(0,j,k) \equiv b_{kj}$, $c(i,j,0) \equiv 0$, $c_{ij} \equiv c(i,j,N_3)$.

Matrix multiplication problem, and consequently, Algorithm_1, is a three-dimensional. According to Algorithm_1 orthogonal two-dimensional (2D) SAs can be synthesized (see, for example [4, 7, 12, 13, 19, 20, 21]). If in Algorithm_1 one of the dimensions $N_1, N_2$ or $N_3$ is equal 1, 2D orthogonal SAs degrade to 1D bidirectional SAs suitable for implementation of matrix-vector product. This fact can be used to compute matrix product on 1D bidirectional SAs iteratively, by repeating the procedure appropriately number of times. Our goal is to design regular BLSA with optimal number of PEs for a given problem size which implements matrix multiplication. Also, the computation time should be minimized for a given number of PEs.

## 3 Mathematical models and systolic algorithms

There are several ways to compute matrix product that can be used to design BLSA:

1) To view it as a product of matrix $A$ and column-vectors of matrix $B$, i.e.

$$C = [\vec{C}_{\cdot 1}\, \vec{C}_{\cdot 2} \ldots \vec{C}_{\cdot N_2}] = [A\vec{B}_{\cdot 1}\, A\vec{B}_{\cdot 2} \ldots A\vec{B}_{\cdot N_2}], \tag{2}$$

where $\vec{B}_{\cdot j}$ and $\vec{C}_{\cdot j}$ are row-vectors of the corresponding matrices. In this case the basic computation of the BLSA would be $\vec{C}_{\cdot 1} = A \cdot \vec{B}_{\cdot 1}$, and the resulting matrix $C$ can be obtained by repeating the computation of this type $N_2$ times.

2) or

$$C = A \cdot B = [\vec{C}_{1\cdot}\, \vec{C}_{2\cdot} \ldots \vec{C}_{N_1\cdot}]^T = [\vec{A}_{1\cdot} B\, \vec{A}_{2\cdot} B \ldots \vec{A}_{N_1\cdot} B]^T\, [\vec{C}_{1\cdot}\, \vec{C}_{2\cdot} \ldots \vec{C}_{N_1\cdot}]^T \tag{3}$$

where $\vec{A}_{i\cdot}$ and $\vec{C}_{i\cdot}$ are column-vectors of the corresponding matrices. In this case basic computation performed by the BLSA is $\vec{C}_{1\cdot} = \vec{A}_{1\cdot} \cdot B$ and it can be viewed as a product of two matrices of dimension $1 \times N_3$ and $N_3 \times N_2$. The resulting matrix $C$ can be obtained by repeating the computation of this type $N_1$ times.

3) and

$$C = \sum_{k=1}^{N_3} C^{(k)} = \sum_{k=1}^{N_3} \vec{A}_{\cdot k} \vec{B}_{k\cdot} \tag{4}$$

In this case basic computation performed by the BLSA is $C^{(1)} = \vec{A}_{\cdot 1} \cdot \vec{B}_{1\cdot}$ and it can be viewed as a product of two matrices of order $N_1 \times 1$ and $1 \times N_2$. The resulting matrix $C$ can be obtained by repeating the computation of this type $N_3$ times.

According to (2), (3) and (4) four bidirectional linear systolic arrays, denoted as SA1, SA2, SA3 and SA4, that compute matrix product can be obtained. Here we will omit the procedure for SA synthesis since it is similar to that described in [2]. Instead, we will give exact formulas for the synthesis of each BLSA that implements corresponding matrix multiplication algorithm.

### 3.1 The array SA1

The first array, SA1, computes $\vec{C}_{\cdot 1} = A \cdot \vec{B}_{\cdot 1}$. The systolic algorithm that corresponds to this computation is is obtained from the Algorithm_1 by substituting $N_2 = 1$ (i.e. for $j = 1$). The array SA1 is obtained for the direction $\vec{\mu} = [1\,0\,1]^T$ as degenerated 2D orthogonal SA (see for example [1], [2], [4], [13], [19], [21]). In order to minimize space parameters of SA1 the systolic algorithm should be adjusted to the projection direction vector $\vec{\mu} = [1\,0\,1]^T$. The algorithm adjusted over index variable $i$ obtained from Algorithm_1 by putting $j = 1$ has the following form

**Algorithm_2**
**for** $k := 1$ **to** $N_3$ **do**
**for** $i := 1$ **to** $N_1$ **do**
$a(i,1,i+k-1) := a(i,0,i+k-1);$
$b(i,1,i+k-1) := b(i-1,1,i+k-1);$

$$c(i,1,i+k-1) := c(i,1,i+k-2) + a(i,1,i+k-1) * b(i,1,i+k-1);$$

where $a(i,0,k+N_3) \equiv a(i,0,k) \equiv a_{ik}$, $b(0,j,k+N_3) \equiv b(0,j,k) \equiv b_{kj}$, $c(i,1,i-1) \equiv c_{i1}^{(0)} \equiv 0$, for each $i = 1,2,\ldots,N_1$ and $k = 1,2,\ldots,N_3$.

Denote by $PE \mapsto \begin{bmatrix} x \\ y \end{bmatrix}$ the PE position in the projection plane. Similarly, denote by $a(i,j,k) \mapsto \begin{bmatrix} x \\ y \end{bmatrix}_a$, $b(i,j,k) \mapsto \begin{bmatrix} x \\ y \end{bmatrix}_b$ and $c(i,j,k) \mapsto \begin{bmatrix} x \\ y \end{bmatrix}_c$ the initial $(x,y)$ positions of elements of the corresponding matrices $A$, $B$ and $C$, respectively.

The array SA1 is obtained according to the following formulas

$$\begin{aligned}
PE \mapsto \begin{bmatrix} x \\ y \end{bmatrix} &= \begin{bmatrix} k-1 \\ 1 \end{bmatrix}, \\
a(i,0,i+k-1) \mapsto \begin{bmatrix} x \\ y \end{bmatrix}_a &= \begin{bmatrix} k-1 \\ 3-2i-k \end{bmatrix} + r_1 \bar{N}_1 \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \\
b(0,1,i+k-1) \mapsto \begin{bmatrix} x \\ y \end{bmatrix}_b &= \begin{bmatrix} 2i+2k-3 \\ 1 \end{bmatrix} + r_1 \bar{N}_1 \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \\
c(i,j,k) \mapsto \begin{bmatrix} x \\ y \end{bmatrix}_c &= \begin{bmatrix} 1-2i \\ 1 \end{bmatrix} + r_1 \bar{N}_1 \begin{bmatrix} 1 \\ 0 \end{bmatrix},
\end{aligned}$$

for $i = 1,2,\ldots,N_1$ and $k = 1,2,\ldots,N_3$, and

$$\bar{N}_1 = \begin{cases} N_1, & \text{if } N_1 \text{ odd} \\ N_1 - 1, & \text{if } N_1 \text{ even} \end{cases}, \tag{5}$$

while $r_1$ is greater of the integers from the set $\{0,1\}$, determined for each $i = 1,2,\ldots,N_1$, that satisfies the inequality

$$-2(i-1) + r_1 \bar{N}_1 < 0, \qquad i = 1 \Longrightarrow r_1 = 0. \tag{6}$$

Parameters $\bar{N}_1$ and $r_1$ are determined such that computation time of the SA1 is minimized.

Data schedule in the array SA1 that implements Algorithm_2 at the beginning of the computation for the case $N_1 = 3$, $N_2 = 2$ and $N_3 = 5$ is presented in Fig. 1.

### 3.2 The array SA2

The array SA2 computes $\vec{C}_{1.}^{T} = (\vec{A}_{1.} \cdot B)^T$. The systolic algorithm that corresponds to this computation is is obtained from the Algorithm_1 by substituting $N_1 = 1$ (i.e. for $i = 1$). The array SA2 is obtained for the direction $\vec{\mu} = [0\,1\,1]^T$ as degenerated 2D orthogonal SA (see for example [1], [2], [4], [13], [19], [21]). In order to minimize space parameters of SA2 the systolic algorithm should be adjusted to the projection direction vector $\vec{\mu} = [0\,1\,1]^T$ (see for example [1], [16], [20]). The algorithm adjusted over index variable $j$ obtained from Algorithm_1 by putting $i = 1$ has the following form

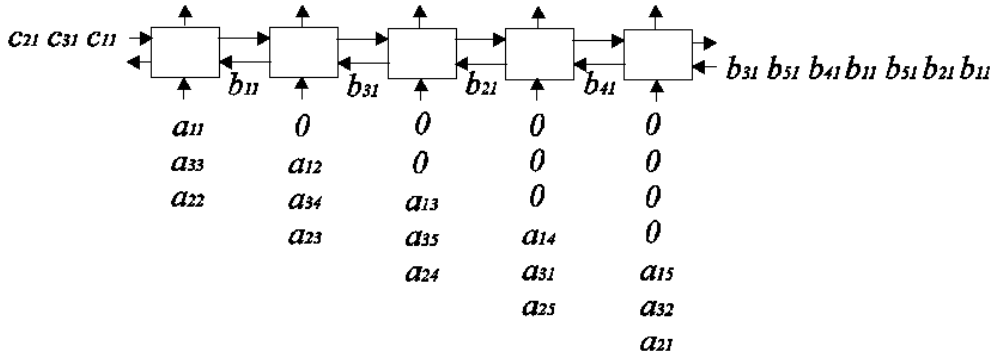


Fig. 1. Data flow in SA1 for the case $N_1 = 3$, $N_2 = 2$ and $N_3 = 5$

**Algorithm_3**
**for** $k := 1$ **to** $N_3$ **do**
**for** $j := 1$ **to** $N_2$ **do**
$a(1, j, k+j-1) := a(1, j-1, k+j-1);$
$b(1, j, k+j-1) := b(0, j, k+j-1);$
$c(1, j, k+j-1) := c(1, j, k+j-2) + a(1, j, k+j-1) * b(1, j, k+j-1);$

where $a(1,0,k+N_3) \equiv a(1,0,k) \equiv a_{1k}$, $b(0,j,k+N_3) \equiv b(0,j,k) \equiv b_{kj}$, $c(1,j,j-1) \equiv c_{1j}^{(0)} \equiv 0$, for each $j = 1,2,\ldots,N_2$ and $k = 1,2,\ldots,N_3$.

The array SA2 is obtained according to the following formulas

$$\begin{aligned}
PE \mapsto \begin{bmatrix} x \\ y \end{bmatrix} &= \begin{bmatrix} k-1 \\ 1 \end{bmatrix} \\
a(1,0,j+l-1) \mapsto \begin{bmatrix} x \\ y \end{bmatrix}_a &= \begin{bmatrix} 2j+2k-3 \\ 1 \end{bmatrix} + r_2 \bar{N}_2 \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \\
b(0,j,j+k-1) \mapsto \begin{bmatrix} x \\ y \end{bmatrix}_b &= \begin{bmatrix} k-1 \\ 3-2j-k \end{bmatrix} + r_2 \bar{N}_2 \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \\
c(1,j,0) \mapsto \begin{bmatrix} x \\ y \end{bmatrix}_c &= \begin{bmatrix} 1-2j \\ 1 \end{bmatrix} + r_2 \bar{N}_2 \begin{bmatrix} 1 \\ 0 \end{bmatrix},
\end{aligned}$$

for $j = 1,2,\ldots,N_2$, $k = 1,2,\ldots,N_3$ and

$$\bar{N}_2 = \begin{cases} N_2, & \text{if } N_2 \text{ odd}, \\ N_2 - 1, & \text{if } N_2 \text{ even} \end{cases} \tag{7}$$

while $r_2$ is greater of the integers from the set $\{0,1\}$, determined for each $j = 1,2,\ldots,N_2$, that satisfies the inequality

$$-2(j-1) + r_2 \bar{N}_2 < 0, \qquad j = 1 \Longrightarrow r_2 = 0. \tag{8}$$

Data schedule in the array SA2 that implements Algorithm_3 at the beginning of the computation, for the case $N_1 = 3$, $N_2 = 2$ and $N_3 = 5$, is presented in Fig. 2.

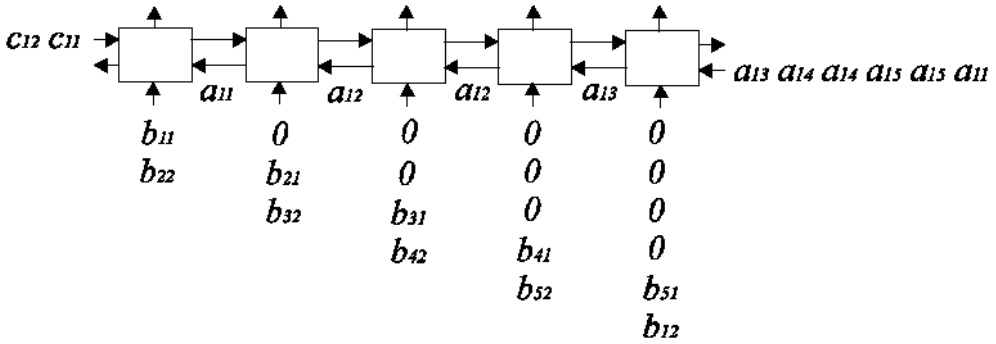


Fig. 2. Data flow in SA2 for the case $N_1 = 3$, $N_2 = 2$ and $N_3 = 5$

### 3.3 The arrays SA3 and SA4

The arrays SA3 and SA4 compute $\vec{C}^{(1)} = \vec{A}_{.1} \cdot \vec{B}_{1.}$. The systolic algorithm that corresponds to this computation is is obtained from the Algorithm_1 by substituting $N_3 = 1$ (i.e. for $k = 1$). The array SA3 is obtained for the direction $\vec{\mu} = [1\,1\,0]^T$ as degenerated 2D orthogonal SA (see for example [1], [2], [4], [13], [19], [21]). In order to minimize space parameters of the BLSA the systolic algorithm should be adjusted to the projection direction vector $\vec{\mu} = [1\,1\,0]^T$ (see for example [1], [16], [20]). In the case of direction $\vec{\mu} = [1\,1\,0]^T$ it is possible to perform adjustment over both index variable $i$ and $j$. The algorithm adjusted over index variable $i$ obtained from Algorithm_1 by putting $k = 1$ has the following form

**Algorithm_4**
**for** $j := 1$ **to** $N_2$ **do**
**for** $i := 1$ **to** $N_1$ **do**
$a(i, i+j-1, 1) := a(i, i+j-2, 1);$
$b(i, i+j-1, 1) := b(i-1, i+j-1, 1);$
$c(i, i+j-1, 1) := c(i, i+j-1, 0) + a(i, i+j-1, 1) * b(i, i+j-1, 1);$

where $a(i, j+N_2, 1) \equiv a(i, 0, 1) \equiv a_{i1}$, $b(0, j+N_2, 1) \equiv b(0, j, 1) \equiv b_{1j}$, $c(i, j, 0) \equiv c_{ij}^{(0)} \equiv 0$, for each $i = 1, 2, \ldots, N_1$ and $j = 1, 2, \ldots, N_2$. The array that implements Algorithm_4 is denoted as SA3.

The array SA3 is obtained according to the following formulas

$$
\begin{aligned}
PE \mapsto \begin{bmatrix} x \\ y \end{bmatrix} &= \begin{bmatrix} j-1 \\ 1 \end{bmatrix} \\
a(i,0,1) \mapsto \begin{bmatrix} x \\ y \end{bmatrix}_a &= \begin{bmatrix} 1-2i \\ 1 \end{bmatrix} + r_1 \bar{N}_1 \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \\
b(0, i+j-1, 1) \mapsto \begin{bmatrix} x \\ y \end{bmatrix}_b &= \begin{bmatrix} 2i+2j-3 \\ 1 \end{bmatrix} + r_1 \bar{N}_1 \begin{bmatrix} -1 \\ 0 \end{bmatrix},
\end{aligned}
$$

$$c(i,i+j-1,0) \mapsto \begin{bmatrix} x \\ y \end{bmatrix}_c = \begin{bmatrix} j-1 \\ 3-2i-j \end{bmatrix} + r_1 \bar{N}_1 \begin{bmatrix} 0 \\ 1 \end{bmatrix},$$

for $i = 1,2,\ldots,N_1$, $j = 1,2,\ldots,N_2$. Parameters $\bar{N}_1$ and $r_1$ are defined by (5) and (6), respectively.

Data schedule in the array SA3 that implements Algorithm_4 at the beginning of the computation, for the case $N_1 = 3$, $N_2 = 2$ and $N_3 = 5$, is presented in Fig. 3.

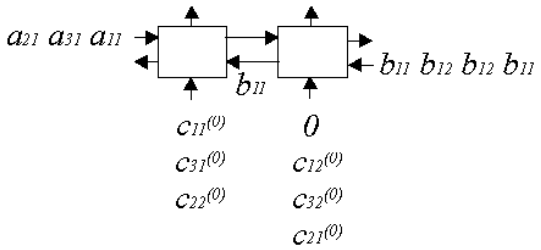


Fig. 3. Data flow in SA3 for the case $N_1 = 3$, $N_2 = 2$ and $N_3 = 5$

The algorithm adjusted over index variable $j$ obtained from Algorithm_1 by putting $k = 1$ has the following form

**Algorithm_5**
**for** $j := 1$ **to** $N_2$ **do**
**for** $i := 1$ **to** $N_1$ **do**
$a(i+j-1,j,1) := a(i+j-1,j-1,1);$
$b(i+j-1,j,1) := b(i+j-2,j,1);$
$c(i+j-1,j,1) := c(i+j-1,j,0) + a(i+j-1,j,1) * b(i+j-1,j,1);$

where $a(i+N_1,0,1) \equiv a(i,0,1) \equiv a_{i1}$, $b(i+N_1,j,1) \equiv b(0,j,1) \equiv b_{1j}$, $c(i+N_1,j,0) \equiv c(i,j,0) \equiv c_{ij}^{(0)} \equiv 0$, for each $i = 1,2,\ldots,N_1$ and $j = 1,2,\ldots,N_2$. The array that implements Algorithm_5 is denoted as SA4.

The array SA4 is obtained according to the following formulas

$$\begin{aligned}
PE \mapsto \begin{bmatrix} x \\ y \end{bmatrix} &= \begin{bmatrix} 1-i \\ 1 \end{bmatrix} \\
a(i+j-1,0,1) \mapsto \begin{bmatrix} x \\ y \end{bmatrix}_a &= \begin{bmatrix} 3-2i-2j \\ 1 \end{bmatrix} + r_2 \bar{N}_2 \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \\
b(0,j,1) \mapsto \begin{bmatrix} x \\ y \end{bmatrix}_b &= \begin{bmatrix} 2j-1 \\ 1 \end{bmatrix} + r_2 \bar{N}_2 \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \\
c(i+j-1,j,0) \mapsto \begin{bmatrix} x \\ y \end{bmatrix}_c &= \begin{bmatrix} 1-i \\ 3-i-2j \end{bmatrix} + r_2 \bar{N}_2 \begin{bmatrix} 0 \\ 1 \end{bmatrix},
\end{aligned}$$

for $i = 1,2,\ldots,N_1$, $j = 1,2,\ldots,N_2$. Parameters $\bar{N}_2$ and $r_2$ are defined by (7) and (8), respectively.

Data schedule in the array SA4 that implements Algorithm_5 at the beginning of the computation, for the case $N_1 = 3$, $N_2 = 2$ and $N_3 = 5$, is presented in Fig. 4.

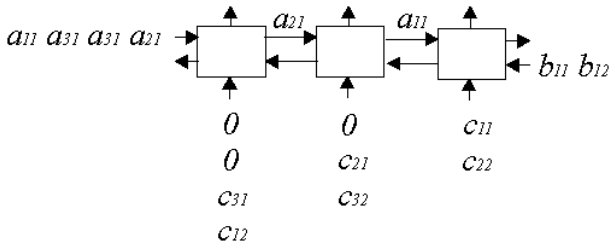


Fig. 4. Data flow in SA4 for the case $N_1 = 3$, $N_2 = 2$ and $N_3 = 5$

## 4 Performance analysis and discussion

In order to compare synthesized arrays, we use following performance measures:

- Number of processing elements, $\Omega$
- Total execution time to compute matrix product, $T_{tot}$
- Efficiency, $E$

The results are summarized in Table 1. All arrays are optimal with respect to a number of processing elements for a given problem size and execution time is minimal possible for that number of processing elements. According to Table 1. we conclude that the efficiency depends on the relation between loop boundaries $N_1$, $N_2$ and $N_3$. When $N_1 > N_3$, the efficiency of the array SA1 is good, but not in the reverse case. Moreover, when $N_3 >> N_1$ the efficiency of SA1 tends to zero, despite the fact that it has optimal number of PEs and that execution time has been minimized. Similarly, the array SA2 has good efficiency when $N_2 > N_3$, bad when $N_3 > N_2$ and tends to zero when $N_3 >> N_2$. This implies that when $N_3 >> \max\{N_1, N_2\}$ one should use the arrays SA3 or SA4 to compute matrix product, since their efficiency do not depend on $N_3$. The overall analysis of the performance measures given in Table 1 brings us to the following conclusion:

- When $N_1 > N_2 > N_3$, the best choice is array SA1;
- When $N_2 > N_1 > N_3$, the best choice is array SA2;
- When $N_1 > N_3 > N_2$ or $N_3 > N_1 > N_2$, the best choice is array SA3;
- When $N_2 > N_3 > N_1$ or $N_3 > N_2 > N_1$, the best choice is array SA4;
- In the square case, i.e. when $N_1 = N_2 = N_3 = N$, the efficiency of all arrays is approximately $\frac{1}{3}$. However, it is not difficult to show that by reordering of initial computations in arrays SA3 and SA4, the execution time can be reduced for the order of $N^2$, and hence improve the efficiency to $E \approx \frac{1}{2}$.

| | SA1 | SA2 | SA3 | SA4 |
|---|---|---|---|---|
| $\Omega$ | $N_3$ | $N_3$ | $N_2$ | $N_1$ |
| $T_{tot} \approx$ | $N_2(N_1+2N_3-2)$ | $N_1(N_2+2N_3-2)$ | $N_3(N_1+2N_2-2)$ | $N_3(N_2+2N_1-2)$ |
| $E \approx$ | $\frac{N_1}{N_1+2N_3-2}$ | $\frac{N_2}{N_2+2N_3-2}$ | $\frac{N_1}{N_1+2N_2-2}$ | $\frac{N_2}{N_2+2N_1-2}$ |

Table 1. Performance measures of the synthesized arrays

## 5 Conclusion

In this paper we have discussed the problem of rectangular matrix multiplication on bidirectional linear systolic arrays. Four different systolic algorithms for computing matrix product were proposed and corresponding BLSA, denoted as SA1, SA2, SA3 and SA4, that implement them were designed. We have compared the arrays SA1, SA2, SA3 and SA4 in term of efficiency. We conclude that the efficiency depends on the relation between loop boundaries $N_1$, $N_2$ and $N_3$.

## References


[1] M. BEKAKOS, E. MILOVANOVIĆ, N. STOJANOVIĆ, T. TOKIĆ, I. MILOVANOVIĆ, I. MILENTIJEVIĆ, *Transformation matrices for systolic array synthesis*, J. Electrotech. Math., Vol. 1, (2002), 9-15.

[2] M. P. BEKAKOS, I. Ž. MILOVANOVIĆ, E. I. MILOVANOVIĆ, T. I. TOKIĆ, M. K. STOJČEV, *Hexagonal systolic arrays for matrix multiplication*, In: Highly Parallel Computations: Algorithms and Applications (M. P. Bekakos, ed.), WITpress, Southampton, Boston, 2001, 175-209.

[3] D. J. EVANS, M. GUŠEV, *The magic of interlocking property: Fast systolic design*, Par. Algor. Appl., No 10, (1997), 195-209

[4] D. J. EVANS, C. R. WAN, *Massive parallel processing for matrix multiplications: a systolic approach*, In: Highly Parallel Computations: Algorithms and Applications (M. P. Bekakos, ed.), WITpress, Southampton, Boston, 2001, 139-173

[5] M. GUŠEV, D. J. EVANS, *Non-linear transformations of the matrix multiplication algorithm*, Inter. J. Comp. Math., No 45, 1992, 1-21

[6] M. Gušev, D. J. Evans, *Procedures for folding transformations*, Inter. J. Comp. Math., No 56, (1995), 19-22

[7] H. V. JAGADISH, T. KAILATH, *A family of new efficient arrays for matrix multiplication*, IEEE Trans. Comput., Vol 38, 1, (1989), 149-155.

[8] H. T. KUNG, *Why systolic architectures?*, Computer, No 15, (1982), 37-46.

[9] H. T. KUNG, C. E. LEISERSON, *Systolic arrays for (VLSI), Introduction to VLSI Systems*, (C. Meed, L. Conway, eds.), Addison-Wesley Ltd., Reading, MA, 1980.

[10] S. Y. KUNG, *VLSI array processors*, Prentice Hall, New Jersey, 1988.

[11] P. LEE, Z. M. KEDEM, *Synthesizing linear array algorithms from nested loop algorithms*, IEEE Trans. Comput., Vol 37, 12, (1988), 1578-1598.

[12] L. MELKEMI, M. TCHUENTE, *Complexity of matrix product on a class of orthogonally connected systolic arrays*, IEEE Trans. Comput., Vol 36, 5, (1987), 615-619.

[13] I. Z. MILENTIJEVIĆ, I. Ž. MILOVANOVIĆ, E. I. MILOVANOVIĆ, M. K. STOJČEV, *The design of optimal planar systolic arrays for matrix multiplication*, Comput. Math. Appl., f Vol 33, 6, (1997), 17-35.

[14] E. I. MILOVANOVIĆ, M. B. TOŠIĆ, I. Ž. MILOVANOVIĆ, I. Z. MILENTIJEVIĆ, ıDesigning processor-time optimal systolic arrays for matrix-vector multiplication, J. Electrotechn. Math., No 1, (1998), 7-19.

[15] I. Ž. MILOVANOVIĆ, E. I. MILOVANOVIĆ, I. Z. MILENTIJEVIĆ, M. K. STOJČEV, *Designing of processor time-optimal systolic arrays for band matrix-vector multiplication*, Comput. Math. Appl., Vol 32, 2, (1996), 21-31.

[16] I. Ž. MILOVANOVIĆ, E. I. MILOVANOVIĆ, T. I. TOKIĆ, I. Z. MILENTIJEVIĆ, N. M. STOJANOVIĆ, *A problem of optimizing space parameters in systolic array deigns*, Proc: Second Conference of Informatics and Information Technology, CiiT'01, (M. Gušev, S. Markowski, eds,), Univ. "ST. Cyril and Methodius", Skopje, 2002, 273-281.

[17] N. M. NOVAKOVIĆ, E. I. MILOVANOVIĆ, M. K. STOJČEV, T. I. TOKIĆ, I. Ž. MILOVANOVIĆ, *Optimization of bidirectional systolic arrays for matrix-vector multiplication*, J. Electrotechn. Math., No 4, (1999), 35-40.

[18] I. V. RAMANKRISHNAN, D. S. FUSSELL, A. SILBERSCHATZ, *Mapping homogenous graphs on linear arrays*, IEEE Trans. Comput., Vol. 35, 3, (1986), 189-209.

[19] S. G. SEDUKHIN, G. Z. KARAPETIAN, *Design of optimal systolic systems for matrix multiplication of different structures*, Report 85, Comput. Center Siberian Division of USSR Academy of Science, Novosibirsk, 1990. (In Russian)

[20] T. I. TOKIĆ, I. Ž. MILOVANOVIĆ, D. M. RANDJELOVIĆ, E. I. MILOVANOVIĆ, *Determining VLSI array size for one class of nested loop algorithms*, In: Advances in Computer and Information Sciences, (U. Güdükbay, T. Dagor, A. Gürsay, E. Gelembe, eds.), IDS Press, 1998, 389-396.

[21] C. R. WAN, D. J. EVANS, *Nineteen ways of systolic matrix multiplication*, Inter. J. Comput. Math., 68, (1998), 39-69.

[22] J. XUE, *Clossed-form mapping conditions for the synthesis of linear processor arrays*, J. VLSI Signal Processing, Vol. 10, 2, (1995), 181-189.

[23] J. XUE, C. LENGAUER, *The synthesis of control signals for one-dimensional systolic arrays*, Integration - The VLSI Journal, Vol 14, 1, (1992), 1-32.