FACTA UNIVERSITATIS (NIŠ)
SER. MATH. INFORM. 12 (1997), 283–296

DESIGNING HEXAGONAL SYSTOLIC ARRAY BY COMPOSITE MAPPINGS*

E. I. Milovanović, G. V. Milovanović, I. Ž. Milovanović and D. Milosavljević

This paper is dedicated to Professor D. S. Mitrinović

Abstract. This paper propose a composite mapping strategy to reduce a number of processing elements (PE) and execution time in hexagonal systolic array (SA) for rectangular matrix multiplication. We prove that sufficient number of PEs in hexagonal SA for multiplication of matrices $A = (a_{ik})_{N_1 \times N_3}$ and $B = (b_{kj})_{N_3 \times N_2}$ is $N_3 \min\{N_1, N_2\}$ and that this number achieves an execution time of $N_1 + N_2 + 2N_3 - 3$ time units. The obtained results outperform those obtained by the standard synthesis procedures. This is achieved without increasing PE or array complexity.

1. Introduction

The ever-increasing demands for speed and performance in a wide range of applications, including telecommunications, signal processing, scientific computing, weather prediction, and real-time process control, clearly point out to a large-scale computation requirement that may be satisfied only by revolutionary supercomputing technology. A very promising trend is the use of VLSI technology for building processor arrays on one chip. This technology has inspired many innovative designs in processors architectures [1–12]. Systolic arrays (SA) are suitable for VLSI implementation because of simple and regular design, and nearest-neighbor interconnections. The systolic array concept was invented by Kung and Leiserson [1]. A systolic system is a network of processing elements (PE) that rhythmically compute and pass data through the system. Once adata item is brought from memory to systolic array, it can be used effectively in each cell as it passes while

Received February 25, 1996.
1991 *Mathematics Subject Classification.* Primary 68M07; Secondary 68Q35.
*This work was partly supported by the Serbian Scientific Foundation under grant #04M03.

being "pumped" from cell to cell along the array. Even if communication rate with the memory is slow compared to the processor rate, the data is processed by several PEs and updated only once, so enhanced processing power is achieved. Triangular, square, and hexagonal planar arrays with four or six nearest-neighbor interconnections in vertical, horizontal, and diagonal directions are the planar arrays usually used.

Most of the early systolic arrays were designed in *ad hoc*, case-by-case manner. Nowadays, one of the most challenging problems in systolic processing is the development of a methodology for mapping an algorithm into systolic architecture. Many such methodologies have been proposed in the last decade [11–29]. Most of these are based on concept of dependence vectors to order in time and space the index points representing the algorithm. The ordered index points are represented by nodes in a dependence graph. The systolic array structure that includes PEs locations and communication links between them, can be obtained simply by projecting the dependence graph into lower dimensional processor space. If more than one valid direction of the projection exists, different designs are obtained.

The common characteristic of the methods based on the above approach is that obtained SAs are not always optimal in the sense of number of PEs and/or execution time. Thus, for example, two-dimensional hexagonal SA for multiplication of two matrices $A = (a_{ik})_{N_1 \times N_3}$ and $B = (b_{kj})_{N_3 \times N_2}$ has $M(P) = N_2N_3 + (N_1 - 1)(N_2 + N_3 - 1)$ PEs, and requires $T_{\text{tot}} = 2N_1 + N_2 + 2N_3 - 4$ time units to perform the multiplication. Here the duration of multiply-add operation is taken as a time unit. In the case of square matrices the corresponding values are $M(P) = 3n^2 - 3n + 1$ and $T_{\text{tot}} = 5n - 4$, where n is a size of matrix. Compared with orthogonal arrays these results are poor.

The objective of this paper is to provide an efficient mapping strategy which will enable us to obtain space-time optimal 2D hexagonal SA for multiplication of rectangular matrices. We will prove that sufficient number of PEs in the 2D hexagonal array is $M(\bar{P}) = N_3 \min\{N_1, N_2\}$ (i.e., $M(\bar{P}) = n^2$) and that this number achieves an execution time of $T_{\text{tot}} = N_1 + N_2 + 2N_3 - 3$ (i.e., $T_{\text{tot}} = 4n - 3$) time units. We will prove that when $N_3 \leq \max\{N_1, N_2\}$ is valid, hexagonal array obtained by the proposed composite mapping has the same number of PEs as the best orthogonal array.

The rest of the paper is organized as follows. Section 2 contains the problem definition. In Section 3 we present our mapping strategy which enables us to obtain space-time optimal hexagonal SA without increasing PE or array complexity. Section 4 contains discussion of the obtained results.

2. Background

Consider the multiplication of matrix $A = (a_{ik})$ of order $N_1 \times N_3$ by matrix $B = (b_{kj})$ of order $N_3 \times N_2$ to give a resulting matrix $C = (c_{ij})$ of order $N_1 \times N_2$, where

$$c_{ij} = \sum_{k=1}^{N_3} a_{ik} b_{kj}, \qquad 1 \le i \le N_1, \qquad 1 \le j \le N_2. \tag{2.1}$$

To compute c_{ij} the following recurrence relation can be used:

$$c_{ij}^{(k)} = c_{ij}^{(k-1)} + a_{ik} b_{kj}, \qquad k = 1, \ldots, N_3, \qquad c_{ij}^{(0)} = 0. \tag{2.2}$$

In a fairly straightforward way, one can obtain regular iterative algorithm that performs the computation (2.2):

```
ALGORITHM_1
 for k := 1 to N3 do
  for j := 1 to N2 do
   for i := 1 to N1 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);
   endfor;
  endfor;
 endfor;
```

with

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

The ALGORITHM_1 enables representation of computation (2.2) in a three-dimensional Euclidean space Z^3. The ordering of computations in ALGORITHM_1 can be described by a dependence graph $\Gamma = \{P_{\text{int}}; E\}$, where $P_{\text{int}} = \{(i,j,k) \mid 1 \le i \le N_1, 1 \le j \le N_2, 1 \le k \le N_3\}$ is a space of inner computations (or iteration space) of the ALGORITHM_1, and $E = \{\vec{e}_b^{\,3}, \vec{e}_a^{\,3}, \vec{e}_c^{\,3}\}$, $\vec{e}_b^{\,3} = (1,0,0)$, $\vec{e}_a^{\,3} = (0,1,0)$, $\vec{e}_c^{\,3} = (0,0,1)$ are local dependence vectors for elements of matrices B, A and C, respectively. The graph Γ can be mapped into the projection plane along some allowable direction $\vec{\mu}$, using corresponding transformation matrix $L(\vec{\mu})$. The well-known 2D hexagonal systolic array [1–11] is obtained by the direction $\vec{\mu} = (1,1,1)$. The corresponding transformation matrix may have the following form

$$L(\vec{\mu}) = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \end{bmatrix}. \tag{2.3}$$

The position of each PE in 2D hexagonal SA is described by its (x, y) coordinates which are obtained according to the following mapping

$$P_{\text{int}} \overset{L(\vec{\mu})}{\longmapsto} P, \tag{2.4}$$

where P is a set of integer points in the projection plane where PEs are located. The (x, y) locations of PEs are determined from the following equation

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} i \\ j \\ k \end{bmatrix} = \begin{bmatrix} i-k \\ j-k \end{bmatrix} \tag{2.5}$$

for $1 \leq i \leq N_1$, $1 \leq j \leq N_2$, $1 \leq k \leq N_3$. The interconnections between the PEs are implemented along the directions $\vec{e}_\gamma^{\,2}$, $\gamma \in \{a, b, c\}$, which are projections of the data dependence vectors $\vec{e}_\gamma^{\,3}$ in the projection plane, i.e.,

$$\vec{e}_\gamma^{\,2} = L(\vec{\mu})\vec{e}_\gamma^{\,3}, \quad \gamma \in \{a, b, c\}. \tag{2.6}$$

Denote by D an arbitrary set of integer points, and by $M(D)$ the number of elements in D. According to (2.5) we have that the number of PEs in the 2D hexagonal SA is

$$M(P) = N_2N_3 + (N_1 - 1)(N_2 + N_3 - 1). \tag{2.7}$$

In the case of square matrices, i.e., when $N_1 = N_2 = N_3 = n$, we have that

$$M(P) = 3n^2 - 3n + 1. \tag{2.8}$$

The total execution time for the realization of ALGORITHM_1 on the obtained SA is

$$T_{\text{tot}} = 2N_1 + N_2 + 2N_3 - 4, \tag{2.9}$$

i.e.,

$$T_{\text{tot}} = 5n - 4, \qquad N_1 = N_2 = N_3 = n. \tag{2.10}$$

An example of hexagonal SA for $N_1 = 3$, $N_2 = 4$, $N_3 = 2$ is shown in Fig. 1. During a clock cycle, each PE receives three data items from three different pipes and executes a multiply-add operation. These data items

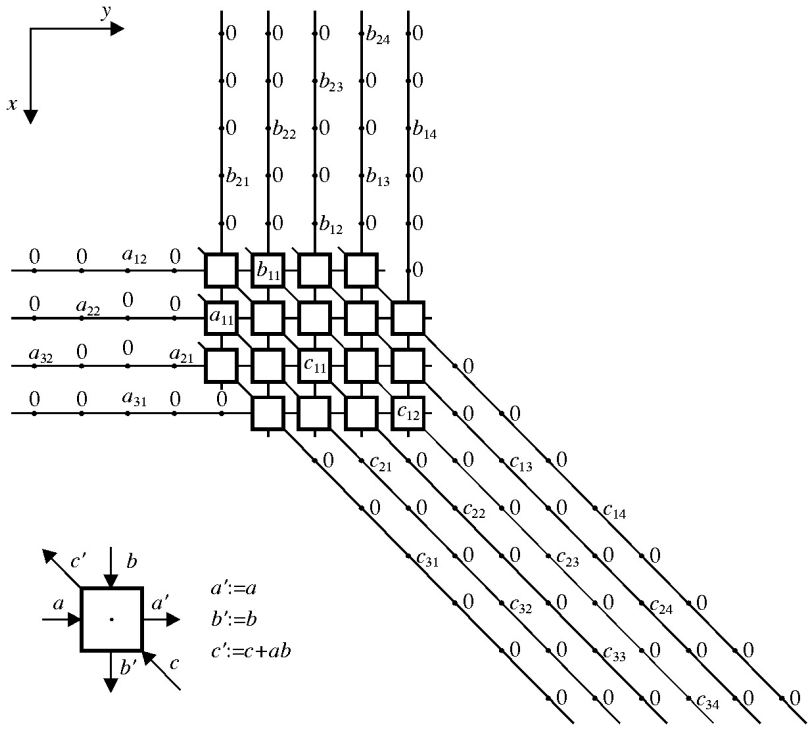

FIG. 1: Hexagonal SA obtained by standard procedure when $N_1 = 3$, $N_2 = 4$, and $N_3 = 2$

advance into neighboring PEs along their own pipes synchronously in the next clock cycle. the efficiency of the array is 1/2 (50%), since only half of the PEs are active in every clock cycle.

If we recall that for square matrix multiplication optimal SA should have n^2 PEs, it is obvious that the array size given by (2.8) is too large. Ref. [30] describes a methodology to design a family of optimal SAs for square matrix multiplication. The procedure given in [30] cannot be easily extended to the case of rectangular matrices. In this paper we describe a new efficient method to design optimal 2D hexagonal SA for multiplication of rectangular matrices.

3. Composite Mapping

Before we formulate our optimal mapping strategy, let us point out to the important feature of the standard procedure that gives hexagonal array with great number of PEs. The graph $\Gamma = \{P_{\text{int}}, E\}$, i.e., the set P_{int}, is

not accommodated to the direction $\vec{\mu} = (1,1,1)$ before mapping (2.4) is applied. Without the accommodation, the set P_{int} is suitable for orthogonal projections only. Namely, all projections, except orthogonal, will give the image P with more than n^2 image pints. Therefore, we are looking for some linear transformation T, which maps set P_{int} into a new set P'_{int} such that when mapping (2.4) is applied on P'_{int} its image $\bar{P}$, has fewer points than P. The idea is pictorially presented in Fig. 2.

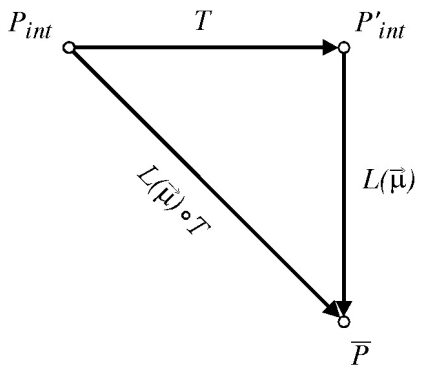

FIG. 2: Illustration of composite mapping

We will show that there are two transformations of type T that are suitable for the direction $\vec{\mu} = (1,1,1)$. So, we will differ between composite mapping (a) and (b).

3.1. Composite mapping (a). Let $T_1 : \quad P_{\text{int}} \mapsto P^1_{\text{int}}$ be a mapping, and let $[i\,j\,k]^T$, $1 \le i \le N_1$, $1 \le j \le N_2$, $1 \le k \le N_3$ be a point from P_{int}, and $[u\,v\,w]^T$ its image in P^1_{int}. The mapping T_1 is defined by

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} i \\ j \\ k \end{bmatrix} + \begin{bmatrix} 0 \\ -1 \\ -1 \end{bmatrix} = \begin{bmatrix} i \\ i+j-1 \\ i+k-1 \end{bmatrix}. \tag{3.1}$$

The mapping (3.1) is a bijection. This statement can easily be verified from the following equivalence

$$\begin{bmatrix} u_1 \\ v_1 \\ w_1 \end{bmatrix} = \begin{bmatrix} u_2 \\ v_2 \\ w_2 \end{bmatrix} \iff \begin{bmatrix} i_1 \\ j_1 \\ k_1 \end{bmatrix} = \begin{bmatrix} i_2 \\ j_2 \\ k_2 \end{bmatrix}.$$

Now, instead of ALGORITHM_1 we introduce a re-indexed algorithm for rectangular matrix multiplication:

ALGORITHM_2
for $k := 1$ **to** N_3 **do**
for $j := 1$ **to** N_2 **do**
for $i := 1$ **to** N_1 **do**
$u := i$;
$v := i + j - 1$;
$w := i + k - 1$;
$a(u, v, w) := a(u, v - 1, w)$;
$b(u, v, w) := b(u - 1, v, w)$;
$c(u, v, w) := c(u, v, w - 1) + a(u, v, w) * b(u, v, w)$;
endfor;
endfor;
endfor;

with the following periodicity of data items

$$
\begin{aligned}
& a(i,j,k) \equiv a(i+N_1,j,k) \equiv a(i,j+1,k) \equiv a(i,j,k+N_3) \equiv a_{ik}, \\
& b(i,j,k) \equiv b(i+1,j,k) \equiv b(i,j+N_2,k) \equiv b(i,j,k+N_3) \equiv b_{kj}, \\
& c(i,j,k) \equiv c(i+N_1,j,k) \equiv c(i,j+N_2,k) \equiv c(i,j,k+N_3) \equiv c_{ij}^{(k)},
\end{aligned} \tag{3.2}
$$

for $1 \le i \le N_1$, $1 \le j \le N_2$, $1 \le k \le N_3$. Instead of graph Γ, we have graph $\Gamma_1 = \{P_{\text{int}}^1, E\}$ which describes the ordering of computations in ALGORITHM_2. If we now apply mapping $Ł(\vec{\mu}) : P_{\text{int}}^1 \mapsto \bar{P}_1$, we will obtain that (x, y) positions of image points are given by

$$
\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} i \\ i+j-1 \\ i+k-1 \end{bmatrix} = \begin{bmatrix} 1-k \\ j-k \end{bmatrix} \tag{3.3}
$$

for $1 \le i \le N_1$, $1 \le j \le N_2$, $1 \le k \le N_3$. Note that (3.3) also determines the (x, y) positions of PEs in the 2D hexagonal SA for realization of ALGORITHM_2. The communication links between neighboring PEs are implemented along the directions defined by (2.6). From (3.3) we conclude that, now, the array size is

$$
M(\bar{P}_1) \;=\; N_2 N_3 \tag{3.4}
$$

processing elements. Required execution time for realization of ALGORITHM_2 is

$$
T_{\text{tot}}^1 \;=\; N_1 + N_2 + 2N_3 - 3. \tag{3.5}
$$

In the case of square matrices, namely when $N_1 = N_2 = N_3 = n$, according to (3.4) and (3.5) we have that

$$M(\bar{P}_1) = n^2, \qquad T^1_{\text{tot}} = 4n - 3. \tag{3.6}$$

An example of the obtained array for $N_1 = 3$, $N_2 = 4$, $N_3 = 2$, is shown in Fig. 3. If we compare the values given by (3.4) and (3.5) with those given by (2.7) and (2.9), the advantage of the proposed composite mapping over existing synthesis methods is obvious. Note that the array efficiency is now 100%, since data streams enter the array in consecutive time moments. this is achieved without increasing processor and/or communication complexity.

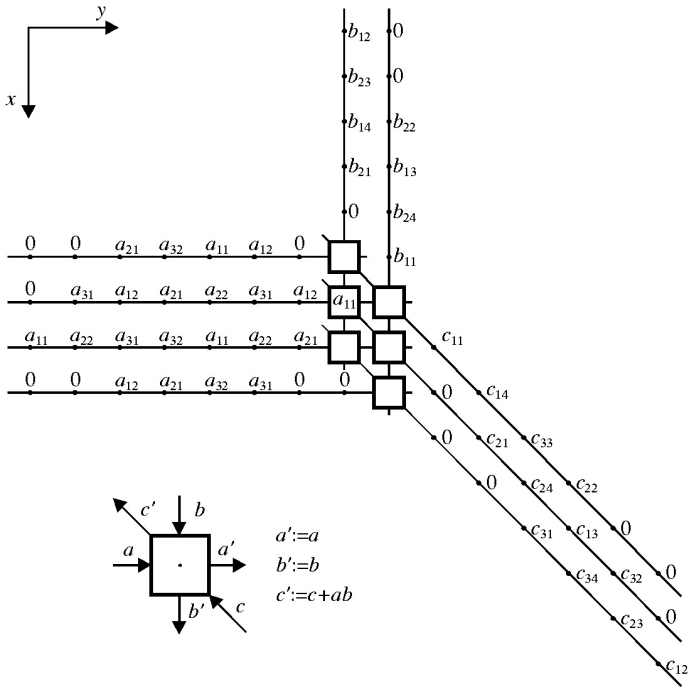

FIG. 3: Hexagonal SA obtained by composite mapping (a) when $N_1 = 3$, $N_2 = 4$, and $N_3 = 2$

3.2. Composite mapping (b). Another possible mapping $T_2 \colon P_{\text{int}} \mapsto \bar{P}^2_{\text{int}}$ is defined by

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} i \\ j \\ k \end{bmatrix} + \begin{bmatrix} -1 \\ 0 \\ -1 \end{bmatrix} = \begin{bmatrix} i+j-1 \\ j \\ j+k-1 \end{bmatrix} \tag{3.7}$$

for $1 \leq i \leq N_1$, $1 \leq j \leq N_2$, $1 \leq k \leq N_3$. The mapping T_2 is a bijection also.

The appropriate re-indexed matrix multiplication algorithm has the following form:

ALGORITHM_3
for $k := 1$ **to** N_3 **do**
for $j := 1$ **to** N_2 **do**
for $i := 1$ **to** N_1 **do**
$u := i + j - 1$;
$v := j$;
$w := j + k - 1$;
$a(u, v, w) := a(u, v - 1, w)$;
$b(u, v, w) := b(u - 1, v, w)$;
$c(u, v, w) := c(u, v, w - 1) + a(u, v, w) * b(u, v, w)$;
endfor;
endfor;
endfor;

with the data periodicity defined by (3.2). The 2D hexagonal SA for implementation of ALGORITHM_3 is completely determined by set $\{\bar{P}_2, E'\}$ where $\bar{P}_2$ is obtained by composite mapping $(L(\vec{\mu}) \cdot T_2) : P_{\text{int}} \mapsto \bar{P}_2$, and $E' = \{\vec{e}_b^2, \vec{e}_a^2, \vec{e}_c^2\}$ being defined by (2.6). The (x, y) positions of image points, i.e., PEs' positions in the projection plane, are given by

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} i + j - 1 \\ j \\ j + k - 1 \end{bmatrix} = \begin{bmatrix} i - k \\ 1 - k \end{bmatrix} \tag{3.8}$$

for $1 \leq i \leq N_1$, $1 \leq j \leq N_2$, $1 \leq k \leq N_3$. The communication links between neighboring PEs are implemented along directions defined by (2.6). According to (3.8) we have that array size is

$$M(\bar{P}_2) = N_1 N_3. \tag{3.9}$$

The execution time for realization of ALGORITHM_3 is given by (3.5), i.e., it is the same as for ALGORITHM_2 implemented on the array obtained by composite mapping (a). An example of the array obtained by composite mapping (b) for $N_1 = 3$, $N_2 = 4$, $N_3 = 2$ is shown in Fig. 4.

Before we discuss the obtained results we shall show that composite mappings (a) and (b) are directly influenced by the direction $\vec{\mu} = (1, 1, 1)$. Since the proof is identical for both case (a) and (b), we will cite a proof for case (a), only.

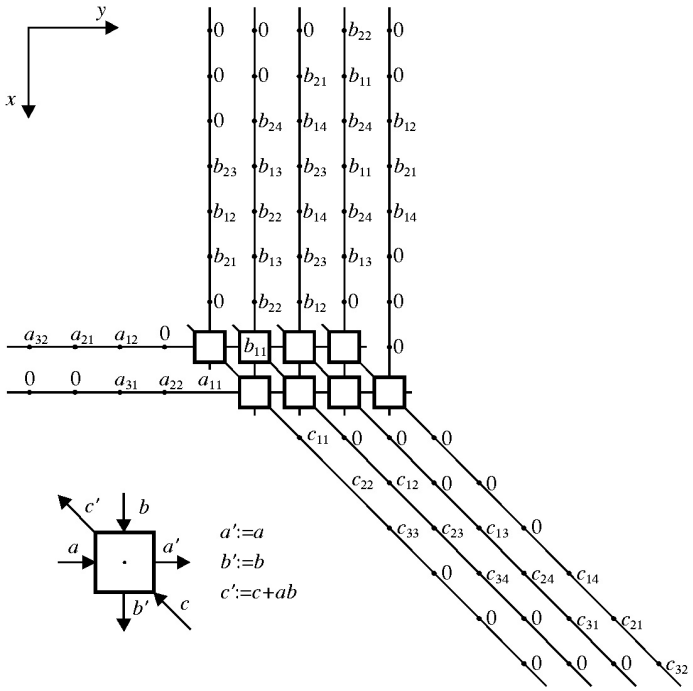

FIG. 4: Hexagonal SA obtained by composite mapping (b) when $N_1 = 3$, $N_2 = 4$, and $N_3 = 2$

According to the equation (3.1) we have that $u = i$, $v = i + j - 1$, $w = i + k - 1$. If we take i as a parameter, we can rewrite preceding equations as

$$\frac{u-1}{1} = \frac{v-j}{1} = \frac{w-k}{1}.$$

On the other hand, the above equations define a set of $N_2 \cdot N_3$ parallel lines with direction $\vec{\mu} = (1, 1, 1)$. Mapping defined by (3.1) requires that each line passes through exactly N_1 points of index set $\bar{P}^1_{\text{int}}$. This fact enables us to obtain considerably better results compared with those find in the literature.

4. Discussion

The main observation about obtained results is that composite mapping gives two hexagonal arrays for multiplication of rectangular matrices. Depending on matrices dimensions N_1 and N_2, we can chose the array with fewer number of PEs. Recall that standard procedures give single hexagonal array. According to composite mappings (a) and (b) we state the following result:

Theorem 1. *The sufficient number of PEs in 2D hexagonal array required for multiplication of rectangular matrices* $A = (a_{ik})$ *of order* $N_1 \times N_3$ *and* $B = (b_{kj})$ *of order* $N_3 \times N_2$ *is*

$$M(\bar{P}) = N_3 \cdot \min\{N_1, N_2\}. \tag{4.1}$$

This number of PEs achieves an execution time of

$$T_{\text{tot}} = N_1 + N_2 + 2N_3 - 3. \tag{4.2}$$

Corollary 1. *According to Theorem 1, for* $N_1 = N_2 = N_3 = n$ *we have that required number of PEs in hexagonal array for square matrix multiplication is*

$$M(\bar{P}) = n^2 \tag{4.3}$$

with execution time

$$T_{\text{tot}} = 4n - 3. \tag{4.4}$$

The result of Corollary 1 was proved in [30].

By comparing the number of PEs and execution time stated by Theorem 1 with values given by (2.7) and (2.9), and results stated by Corollary 1 with (2.8) and (2.10), it is obvious that results obtained by composite mapping are considerable better then those obtained by existing synthesis procedures.

Now, let us compare the number of PEs and execution time of hexagonal array obtained by composite mapping with the corresponding values for orthogonal arrays obtained by standard procedures. Recall that by orthogonal projections three arrays can be obtained. The main feature of these arrays is that some data variables are resident in the array, i.e., they are not fully pipelined. This is the main drawback of these arrays. Namely, from the fault-tolerance aspect it is desired that all data variables be propagated from one PE to the other. In other words, all data variables should be accessible and should not be stored in the PEs [24]. The main advantage of orthogonal arrays over hexagonal, were less number of PEs and shorter execution time.

In the best case, the number of PEs in the orthogonal array is equal to

$$M(\hat{P}) = \min\{N_1N_2, N_1N_3, N_2N_3\}. \tag{4.5}$$

According to (4.1) and (4.5) it can be concluded that when

$$N_3 \leq \max\{N_1, N_2\} \tag{4.6}$$

is valid, hexagonal array obtained by composite mapping has the same number of PEs as the best orthogonal array. Note that when $N_1 = N_2 = N_3 = n$, orthogonal arrays always have n^2 PEs, as well as hexagonal array obtained by composite mapping.

When

$$N_3 > \max\{N_1, N_2\} \tag{4.7}$$

is valid, the number of PEs in hexagonal array obtained by composite mapping is greater than that of the best orthogonal array. The case when inequality (4.7) is valid is recognized in the literature as problem size anomaly. This problem was considered in [31] for the case of matrix-vector multiplication. The approach used in [31] increases the PE's complexity. This problem is beyond the scope of this paper.

REFERENCES

1. H. T. Kung and C. E. Leiserson: *Systolic arrays for (VLSI).* In: Introduction to VLSI Systems (C. Mead, L. Conway, eds.), Addison Wesley, Reading, MA, 1980.
2. K. Hwang and F. A. Briggs: *Computer Architecture and Parallel Processing.* McGraw-Hill, New York, 1984.
3. V. V. Voevodin: *Mathematical models and methods in parallel processing.* Nauka, Moscow, 1986 (Russian).
4. S. Y. Kung: *VLSI Array Processors.* Prentice Hall, Englewood Cliffs, N.J., 1988.
5. S. Akl: *The Design and Analysis of Parallel Algorithms.* Prentice Hall, Englewood Cliffs, N.J., 1989.
6. K. M. Chandy and J. Misra: *Parallel Program Design.* Addison Wesly Publishing, Reading, Massachusetts, 1989.
7. J. H. Moreno and T. Lang: *Matrix Computations on Systolic Type Arrays.* Kluwer Academic Publishers, Boston – London – Dorchrecht, 1992.
8. D.I. Moldovan: *Parallel Processing: From Applications to Systems.* Morgan Kaufman Publishers, San Mateo, 1993.
9. G. S. Almasi and A. Gottlieb: *Highly Parallel Computing.* The Benjamin/Cammings Publishing Co., Inc., Redwood City, California, 1994.
10. M. R. Zorgham: *Computer Architecture.* Prentice Hall, International Editions, 1996.
11. H. B. Lee and R. O. Grondin: *A comparison of systolic architectures for matrix multiplications.* IEEE J. Solid-State Circ. **23** (1988), 285–289.
12. H. T. Kung: *Why systolic architectures?.* Computer **15** (1982), 37–46.
13. D.I. Moldovan: *On the design of algorithms for VLSI systolic arrays.* Proc. IEE **71** (1983), 113–120.

14. D.I. Moldovan and J.A.B. Fortes: *Partitioning of algorithms for fixed size VLSI architectures.* IEEE Trans. Comput. **C-35** (1) (1986), 1–12.

15. J.A.B. Fortes, K.S. Fu, and B.W. Wah: *Systematic design approaches for algorithmically specified systolic arrays.* In: Computer Architecture (V. Milutinović, ed.) Elsevier Ltd., New York, 1988, pp. 454–494.

16. G.-J. Li and B.i W. Wah: *The design of optimal systolic arrays.* IEEE Trans. Comput. **C-34** (1) (1985), 66–77.

17. S.-Y. Kung, K.S. Arun, R.J. Gal-Ezer, and D.V.B. Rao: *Wavefront array processor: Language, architecture and applications.* IEEE Trans. Comput. **C-34** (11) (1982), 1054–1065.

18. W.L. Miranker and and A. Winkler: *Space-time representations of computational structures.* Computing **32** (1984), 93–114.

19. K.H. Cheng, S. Sahni: *VLSI systems for band matrix multiplication.* Parallel Comput. **4** (1987), 239–258.

20. P. Lee and Z.M. Kedem: *Synthezing linear array algorithms from nested for loop algorithms.* IEEE Trans. Comput. **37** (12) (1988), 1578–1598.

21. H.V. Jagadish and T. Kailath: *A family of new efficient arrays for matrix multiplication.* IEEE Trans. Comput. **38** (1) (1989), 149–155.

22. C. Lengauer: *A view of systolic design.* In: Proc. International Conference Parallel Computing Technologies (N.N. Mirenkov, ed.), Novosibirsk, World Scientific, Singapore, 1991, pp. 32–46.

23. C.-W. Jen and D.-M. Kwai: *Data flow representation of iterative algorithms for systolic arrays.* IEEE Trans. Comput. **41** (3) (1992), 351–355.

24. M. Gušev and D.J. Evans: *Nonlinear transformations of the matrix multiplication algorithm.* Inter. J. Comput. Math. **45** (1992), 1–21.

25. M.O. Esonu, A.J. Al-Khalili, S. Hariri, and D. Al-Khalili: *Systolic arrays: how to choose them.* Proc. IEE **E-139** (3) (1992), 179–188.

26. H. Barada and A. El-Amawy: *A methodology for algorithm regularization and mapping into time-optimal VLSI arrays.* Parallel Comput. **19** (1993), 33–61.

27. J.-G. Tsay and Y.-C. Hou: *Generating function and equivalent transformation for systolic arrays.* Parallel Comput. **10** (1989), 347–356.

28. S.G. Sedukhin and G.Z. Karapetian: *Design of Optimal Systolic Systems for Matrix Multiplication of Different Structures.* Report 855, Comp. Center Siberian Division of USSR Academy of Science, Novosibirsk, 1990 (Russian).

29. S.G. Sedukhin: *The designing and analysis of systolic algorithms and structures.* Programming **2** (1991), 20–40 (Russian).

30. I.Z. Milentijević, I. Ž. Milovanović, E.I. Milovanović, and M.K. Stojčev: *The design of optimal planar systolic arrays for matrix multiplication.* Comput. Math. Appl. **3** (6) (1997), 17–35.

31. Y.-C. Lin: *Array size anomaly of problem-size independent systolic arrays for matrix-vector multiplication.* Parallel Comput. **17** (1991), 515–522.

32. I. Ž. Milovanović, E.I. Milovanović, I.Z. Milentijević, and M.K. Stojčev: *Designing of processor-time optimal systolic arrays for band matrix-vector multiplication.* Comput. Math. Appl. **32** (2) (1996), 21–31.

Faculty of Electronic Engineering
Department of Computer Sciences and Informatics
18000 Niš, Serbia, Yugoslavia
e–mail: ema@elfak.ni.ac.yu

Faculty of Electronic Engineering
Department of Mathematics, P.O. Box 73
18000 Niš, Serbia, Yugoslavia
e–mail: grade@elfak.ni.ac.yu

Faculty of Electronic Engineering
Department of Mathematics, P.O. Box 73
18000 Niš, Serbia, Yugoslavia
e–mail: igor@elfak.ni.ac.yu

Faculty of Electronic Engineering
Department of Computer Sciences and Informatics
18000 Niš, Serbia, Yugoslavia
e–mail: dejanm@elfak.ni.ac.yu