The Design of Optimal Planar Systolic Arrays for Matrix Multiplication

I. Z. MILENTJEVIĆ, I. Ž. MILOVANOVIĆ, E. I. MILOVANOVIĆ AND M. K. STOJČEV

University of Niš, Faculty of Electronic Engineering
P.O. Box 73, Beogradska 14, 18000 Niš, Serbia

(Received October 1996; accepted December 1996)

Abstract—The objective of this paper is to provide a systematic methodology for the design of space-time optimal pure planar systolic arrays for matrix multiplication. The procedure is based on data dependence approach. By the described procedure, we obtain ten different systolic arrays denoted as S₁ to S₁₀ classified into three classes according to interconnection patterns between the processing elements. Common properties of all systolic array designs are: each systolic array consists of \( n^2 \) processing elements, near-neighbour communications, and active execution time of \( 3n - 2 \) time units. Compared to designs found in the literature, our procedure always leads to systolic arrays with optimal number of processing elements. The improvement in space domain is not achieved at the cost of execution time or PEs complexity. We present mathematically rigorous procedure which gives the exact ordering of input matrix elements at the beginning of the computation. Examples illustrating the methodology are shown.

Keywords—Matrix multiplication, Data dependency, Systolic arrays, Mapping.

1. INTRODUCTION

Special purpose computers are used to do computationally intensive tasks occurring frequently in signal processing and numerical computation, in which the algorithms are often regularly structured and inherently parallel. To make use of the parallelism, algorithms are directly mapped into the hardware such as systolic arrays. Systolic architectures are suitable for VLSI implementation because their reliance on nearest-neighbour interconnections and regularity. One common computation intensive task in which systolic architectures are effective is matrix multiplication.

Most of the early systolic arrays were designed in an 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 a systolic architecture. Many such methodologies have been proposed in the last decade [1–20]. 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 with global dependencies and then this graph is transformed into the directed graph with local dependencies. The systolic array structure that includes PEs locations and communication links between PEs can be obtained simply by projecting the dependence graph onto a 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 the same dependence graph is used for each
allowable direction of the projection. As a consequence, the obtained arrays are not always optimal.

There have been several works on how to synthesize optimal systolic array architectures, with each work concentrating on certain optimization criterion, such as smallest number of PEs used, or minimum execution time. Moldovan and Fortes [1], Miranker and Winkler [2] worked on how to minimize the computation time of a systolic array. In addition to minimizing the computation time, Fortes [21] proposed a heuristic approach for optimizing the hardware cost. The array size, which is defined as the number of processors in the array, obviously determines the basic hardware cost. Therefore, a systolic array which has the minimum number of processors gives the optimal solution with respect to this cost function [3].

Our objective is to provide a systematic methodology for synthesizing space-time optimal systolic arrays for matrix multiplication. This is accomplished by forming a periodical and unconfined computation space which is then used for deriving a separate dependence graph, with minimal number of nodes, for each allowable direction of the projection. The later enables that each projection ray passes through the maximal number of nodes of the graph providing that the obtained 2-D systolic array has optimal number of PEs. The procedure proposed in this paper is an improved version of the data dependence method, based on linear transformations, proposed in [4,6]. By our procedure, we obtain ten different planar systolic arrays for matrix multiplication denoted by $S_1$ through $S_{10}$. The common properties of all systolic arrays are: each array consists of $n^2$ processing elements ($n$ is a dimension of square matrix), near-neighbour communications, and active execution time of $3n - 2$ time units. Compared to the designs known from the literature, our procedure always gives systolic arrays with optimal number of processing elements. The improvement in space domain is not achieved at the cost of execution time or PEs complexity.

The rest of the paper is organized as follows. Section 2 contains the problem definition. In Section 3, we present the modification of the standard procedure which enables us to design the space-time optimal SAs. First, we introduce an unconfined computational space. Then, we determine all allowable direction of the projections which give planar SAs. After that, we perform space and time optimization and define the real algorithm for matrix multiplication for each allowable direction of the projection. Section 4 contains a survey of the planar SAs obtained by the proposed procedure. Section 5 contains the discussion of the obtained results and comparison with the known results. Section 6 is a conclusion.

2. PROBLEM DEFINITION

Consider the multiplication of square matrices $A = (a_{ik})$ and $B = (b_{kj})$ of order $n \times n$ to give a resulting matrix $C = (c_{ij})_{n \times n}$, where

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}, \quad i, j = 1, 2, \ldots, n. \quad (1)$$

To compute $c_{ij}$ the following recurrence relation:

$$c_{ij}^{(k)} = c_{ij}^{(k-1)} + a_{ik} b_{kj}, \quad k = 1, \ldots, n, \quad c_{ij}^{(0)} = 0, \quad (2)$$

can be used.

In a fairly straightforward way, one can obtain a regular iterative algorithm that performs the desired computation (2).

Algorithm_1

for $k := 1$ to $n$ do
  for $j := 1$ to $n$ do
    for $i := 1$ to $n$ do
Matrix Multiplication

\[ a(i,j,k) := a(i,j - 1,k) \]
\[ b(i,j,k) := b(i - 1,j,k) \]
\[ c(i,j,k) := c(i,j,k - 1) + a(i,j,k) \ast b(i,j,k) \]

with
\[ a(i,0,k) = a_{ik}, \quad b(0,j,k) = b_{kj}, \quad c(i,j,0) = c_{ij}^{(0)} = 0. \quad (3) \]

The Algorithm 1 enables representation of the computation of matrix C in a three-dimensional Euclidian space \( \mathbb{Z}^3 \), with \( i(1,0,0), j(0,1,0), \) and \( k(0,0,1) \) being basis vectors of the space axes. With \( a(i,j,k) \equiv p_a(i,j,k), b(i,j,k) \equiv p_b(i,j,k), c(i,j,k) \equiv p_c(i,j,k) \), we will denote the positions of \( a_{ik}, b_{kj}, \) and \( c_{ij} \) in \( \mathbb{Z}^3 \), respectively. Note that most of the papers [1-20] dealing with designing of 2-D systolic arrays (SA) for matrix multiplication start from the recurrence (2), i.e., Algorithm 1. As a consequence, the obtained SAs are not always optimal. We consider the optimality through the number of processing elements (PEs) and computation time. In order to obtain space-time optimal SAs for matrix multiplication, we start from the equation
\[ c_{ij}^{(k)} = c_{ij}^{(0)} + a_{ik}b_{kj}, \quad c_{ij}^{(0)} = 0, \quad i, j, k \geq 1. \quad (4) \]

Notice that (4) does not compute the element \( c_{ij} \), but the partial products that constitute the resulting element, only.

For \( a_{ik}, b_{kj}, c_{ij}^{(k)} \), i.e., their space positions \( a(i,j,k), b(i,j,k), c(i,j,k) \), we involve the following periodicity:
\[ a(i,j,k) \equiv a(i + n,j,k) \equiv a(i,j,k + n) = a_{ik}, \]
\[ b(i,j,k) \equiv b(i,j + n,k) \equiv b(i,j,k + n) = b_{kj}, \]
\[ c(i,j,k) \equiv c(i + n,j,k) \equiv c(i,j,k + n) = c_{ij}^{(k)}. \quad (5) \]

According to (4) and (5), the same partial products are obtained periodically in \( \mathbb{Z}^3 \). This represents the crucial novelty in our procedure in respect to the standard ones. Namely, later in the design procedure, we will choose those partial products that are the most suitable for obtaining the optimal SA for the particular direction of the projection.

According to (3), (4), and (5), we construct the algorithm that computes the partial products

**Algorithm 2**

\[
\text{for } k \geq 1 \text{ do}
\]
\[
\text{for } j \geq 1 \text{ do}
\]
\[
\text{for } i \geq 1 \text{ do}
\]
\[
 a(i,j,k) := a(i,j - 1,k)
\]
\[
 b(i,j,k) := b(i - 1,j,k)
\]
\[
 c(i,j,k) := c(i,j,0) + a(i,j,k) \ast b(i,j,k).
\]

This is an unconfined and periodical algorithm. The Algorithm 2 will be used as a suitable basis for obtaining the real algorithm for matrix multiplication for each allowable direction of the projection, separately.

### 3. THE PROCEDURE

#### 3.1. Determining of Initial and Inner Computation Spaces

According to (3), (4), and (5), we will determine in \( \mathbb{Z}^3 \) the spaces of initial computations \( \bar{P}_{\text{in}} \), and inner computations \( \bar{P}_{\text{int}} \).

The space of initial computations \( \bar{P}_{\text{in}} \) is the union of subspaces of initial computations for matrices \( A, B, \) and \( C \), i.e., \( \bar{P}_{\text{in}}(a), \bar{P}_{\text{in}}(b), \) and \( \bar{P}_{\text{in}}(c) \), respectively. According to (3), \( \bar{P}_{\text{in}}(a) \) is a set of all integer points in plane \( j = 0 \ (i, k > 0) \), \( \bar{P}_{\text{in}}(b) \) is a set of integer points in plane \( i = 0 \ (j, k > 0) \), while \( \bar{P}_{\text{in}}(c) \) is a set of integer points in plane \( k = 0 \ (i, j > 0) \).
With \( p_\gamma(i, j, k) \equiv \gamma(i, j, k), \gamma \in \{a, b, c\} \), we will denote either an integer point or its position vector in space \( \mathcal{P}_{\text{in}}(\gamma) \). Similarly, \( p(i, j, k) \) will denote either an integer point or its position vector in \( \mathbb{Z}^3 \).

The space \( \mathcal{P}_{\text{int}} \) is a set of integer points from the first octant of \( \mathbb{Z}^3 \), where partial products defined by Algorithm 2 are obtained.

Figure 1 shows parts of spaces \( \mathcal{P}_{\text{in}} \) and \( \mathcal{P}_{\text{int}} \) in some fixed plane \( k \) for \( n = 4 \), denoted as \( \mathcal{P}_{\text{in}}^{(k)} \) and \( \mathcal{P}_{\text{int}}^{(k)} \). With \( e_3^a = (0, 1, 0) \), \( e_6^b = (1, 0, 0) \), and \( e_7^c = (0, 0, k) \), data flow vectors for elements of matrices \( A, B, \) and \( C \) are denoted. The ordering of computations in Algorithm 2 can be described by a dependence graph \( \Gamma = \{ \mathcal{P}_{\text{int}}, E \} \), where \( E = \{ e_3^a, e_6^b, e_7^c \} \). This graph is periodical and unconfined. Figure 1 also depicts a part of \( \Gamma \) in some fixed plane \( k \).

![Image](image_url)

**Figure 1.** A part of \( \mathcal{P}_{\text{in}}, \mathcal{P}_{\text{int}}, \) and \( \Gamma \) in some fixed plane \( k \) for \( n = 4 \).

### 3.2. Space Optimization

Having obtained graph \( \Gamma = \{ \mathcal{P}_{\text{int}}, E \} \), we can derive the real algorithm for computing matrix product for each allowable direction of the projection, separately. But first, we have to determine all allowable directions of the projections \( \mu(\mu_1, \mu_2, \mu_3) \). Since we are interested in planar SAs, we have to involve the constraint for \( \mu_i \), i.e., \( \mu_i \) can take values only from the set \( \{-1, 0, 1\} \). Thus, the number of allowable directions is 10, namely \( \mu(1, 0, 0), \mu(0, 1, 0), \mu(0, 0, 1), \mu(1, 1, 0), \mu(1, 0, 1), \mu(0, 1, 1), \mu(0, -1, 1), \mu(-1, 1, 1), \mu(-1, -1, 1) \). (see [4,5]).

The systolic arrays obtained by the above directions will be denoted with \( S_1 \) to \( S_{10} \), respectively.

In order to derive the finite space of inner computations \( \mathcal{P}_{\text{in}}(\mu) \) from \( \mathcal{P}_{\text{in}} \), we have to define the corresponding mapping \( \mathcal{P}_{\text{in}} \rightarrow \mathcal{P}_{\text{in}}(\mu) \) for each direction \( \mu \), separately. The mapping is performed as follows. Let \( \mu(\mu_1, \mu_2, \mu_3) \) be an arbitrary direction. Through space points \((1, j, k)\) we set straight lines with the direction \( \mu_1 \mu_2 \mu_3 \). The straight line equations are given by

\[
\frac{u - i}{\mu_1} = \frac{v - j}{\mu_2} = \frac{w - k}{\mu_3}.
\]

The corresponding parameter equations for \( \mu_1 \neq 0 \), are

\[
u = u(i) = i, \quad v = v(i, j) = j + \frac{\mu_2}{\mu_1} (i - 1), \quad w = w(i, k) = k + \frac{\mu_3}{\mu_1} (i - 1),
\]

while for \( \mu_1 = 0 \), we have

\[
u = u(i) = i, \quad v = v(j) = j, \quad w = w(j, k) = \begin{cases} k, & \mu_2 = 0, \\ k + \frac{\mu_3}{\mu_2} (j - 1), & \mu_2 \neq 0. \end{cases}
\]
When (8) is used instead of \((1, j, k)\) in (6), we take points \((i, 1, k)\). Indices \(i, j, k\) are in the range 1 to \(n\), i.e.,

\[
\begin{align*}
\text{for } k & := 1 \text{ to } n \\
\text{for } j & := 1 \text{ to } n \\
\text{for } i & := 1 \text{ to } n.
\end{align*}
\]

(9)

**Remark 1.** Since we have assumed that \(P_{\text{int}}\) is in the first octant, it has to be checked if all \(n^2\) straight lines defined by (6) pass through exactly \(n\) points of \(P_{\text{int}}\). The condition for this is that \(u > 0, v > 0, w > 0\), for all \(i, j, k \in \{1, \ldots, n\}\). When this is not satisfied, and that is for the directions \(\mu(1, 1, -1), \mu(1, -1, 1), \text{ and } \mu(-1, 1, 1)\), the borders (9) have to be modified. This is an elemental modification and it is possible due to periodicity in the space \(P_{\text{int}}\).

According to (7), (8), (9), and Remark 1, we have defined the mapping \(P_{\text{int}} \rightarrow \tilde{P}_{\text{int}}(\mu)\), for each direction \(\mu(\mu_1, \mu_2, \mu_3)\). The same mapping is applied on \(P_{\text{in}}\) to obtain \(\tilde{P}_{\text{in}}(\mu)\). Each space \(\tilde{P}_{\text{int}}(\mu)\) is composed of exactly \(n^3\) integer points. Let us note that each of the \(n^2\) parallel lines, defined by (7) (i.e., (8)), passes through \(n\) points of \(\tilde{P}_{\text{int}}(\mu)\). This feature provides that image of \(\tilde{P}_{\text{int}}(\mu)\), obtained by the mapping, always contains exactly \(n^2\) nodes. As a consequence, each of the synthesized 2-D SAs will be composed of optimal number of processing elements, i.e., \(n^2\).

Having determined the real space of inner computations \(\tilde{P}_{\text{int}}(\mu)\), we can define the real algorithm for matrix multiplication for each direction of the projection as

**Algorithm 3**

\[
\begin{align*}
\text{for } k & := k_1 \text{ to } k_1 + n - 1 \\
\text{for } j & := j_1 \text{ to } j_1 + n - 1 \\
\text{for } i & := i_1 \text{ to } i_1 + n - 1 \\
& a(u, v, w) = a(u, v - 1, w) \\
& b(u, v, w) = b(u - 1, v, w) \\
& c(u, v, w) = c(u, v, w - 1) + a(u, v, w) \ast b(u, v, w),
\end{align*}
\]

where \(u, v, w\) are defined by (7), i.e., (8). The values for \(i_1, j_1, k_1\) are determined by (9) or in accordance with Remark 1. For the initial values and data periodicity, the relations (3) and (5) are still valid. Each real algorithm is associated with the corresponding local dependence graph \(\tilde{G}(\mu) = (\tilde{P}_{\text{int}}(\mu), E)\), \(E = \{v_1, v_2, v_3\}\), where \(v_3 = (0, 0, 1)\). Each node of the local dependence graph \(\tilde{G}(\mu)\) can be viewed as a processing element (PE) performing add-multiply operation. In other words, we have obtained a three-dimensional systolic array (3-D SA) with \(n^3\) PEs for the realization of Algorithm 3.

### 3.3. Time Optimization

Since we have performed the spatial optimization, we are going to consider the optimization in time domain. Therefore, we have to determine the timing function \(t(p)\) which defines temporal distribution of the computation. The timing function is of the form \(t(p) = u + v + w + \beta\), where \(\beta\) is a constant determined from the condition \(t(p_{\text{min}}) = 0\), where \(p_{\text{min}} \in \tilde{P}_{\text{int}}(\mu)\) is a point where the first computation should be performed.

According to (9) for the directions \(S_1\) to \(S_7\), we have that \(p_{\text{min}} = (1, 1, 1)\). so in these cases \(t(p)\) is equal to

\[
t(p) = u + v + w - 3.
\]

(10)

For the design \(S_8\) we have that \(p_{\text{min}} = (1, 1, n)\), for \(S_9\) \(p_{\text{min}} = (1, n, 1)\), and for \(S_{10}\) \(p_{\text{min}} = (n, 1, 1)\), so the corresponding \(t(p)\) is

\[
t(p) = u + v + w - n - 2.
\]

(11)
Let

\[ p_1 = \left( i_1, j + \frac{\mu_2}{\mu_1} (i_1 - 1), k + \frac{\mu_3}{\mu_1} (i_1 - 1) \right) \]

and

\[ p_2 = \left( i_1 + 1, j + \frac{\mu_2}{\mu_1} i_1, k + \frac{\mu_3}{\mu_1} i_1 \right) \]

be two neighbouring points on the line defined by (6). For each of the designs \( S_i, i = 1, \ldots, 10 \), we have to check if \( |\Delta t(p)| = |t(p_2) - t(p_1)| \) is greater or equal to one, i.e., we have to compute

\[ |\Delta t(p)| = |t(p_2) - t(p_1)| = \left| 1 + \frac{\mu_2}{\mu_1} + \frac{\mu_3}{\mu_1} \right| = |\mu_1 + \mu_2 + \mu_3|. \]  

(12)

If \( |\Delta t(p)| > 1 \), the nodes laying on the same line have to be reordered so that the distance one is achieved. We call this reordering a compression. By the compression the space \( \hat{P}_{\text{int}}(\mu) \) is mapped into space \( P_{\text{int}}(\mu) \).

According to (12), we conclude that for designs \( S_1, S_2, S_3, S_5, S_9 \), and \( S_{10} \) hold \( |\Delta t(p)| = 1 \), so there is no need for reordering of the corresponding spaces \( \hat{P}_{\text{int}}(\mu) \), i.e., \( \hat{P}_{\text{int}}(\mu) = P_{\text{int}}(\mu) \).

For the designs \( S_4, S_6, S_8 \) we have that \( |\Delta t(p)| = 2 \), whereas for \( S_7 \) \( |\Delta t(p)| = 3 \). In order to optimize the computational time, the corresponding spaces \( \hat{P}_{\text{int}}(\mu) \) have to be compressed, such that \( |\Delta t(p)| = 1 \) is achieved. As we have already mentioned, the line defined by (6) contains exactly \( n \) different points of space \( P_{\text{int}}(\mu) \). This means that the maximal displacement of the node along the line (6), during the compression can be \((|\mu_1 + \mu_2 + \mu_3| - 1) \cdot n\).

Denote with

\[ F = \{0, 1, |\mu_1 + \mu_2 + \mu_3| - 1\} \]

(13)
a set of displacement factors. The space \( \hat{P}_{\text{int}}(\mu) \) is obtained by translating points of \( P_{\text{int}}(\mu) \) along the line (6) for the displacement \( r\tilde{n} \), where \( r \in F \) and \( \tilde{n} \in \{n - 1, n\} \). During the compression some of the points are moved while the others stay at their positions. There are two things that we should take care of during this compression:

(a) two computational points must not be overlapped; and
(b) the timing of any point cannot be less then timing of point \( p^{\min} \).

To avoid the above pitfalls, let us perform the following analysis. Without deterioring generality assume that \( \mu_1 \neq 0 \) and \( \mu_1 + \mu_2 + \mu_3 > 0 \). Let

\[ p_3 = \left( i_3, j + \frac{\mu_2}{\mu_1} (i_3 - 1), k + \frac{\mu_3}{\mu_1} (i_3 - 1) \right) \]

and

\[ p_4 = \left( i_4, j + \frac{\mu_2}{\mu_1} (i_4 - 1), k + \frac{\mu_3}{\mu_1} (i_4 - 1) \right) \],

be two arbitrary points on the line defined by (6) in the space \( \hat{P}_{\text{int}}(\mu) \). Suppose that the point \( p_4 \) is moved along the line and that it coincides now with \( p_3 \). In that case, the following is valid:

\[ t(p_3) = t(p_4) + \tilde{n}. \]

This can happen when \((i_3 - i_4)\mu_1 (\mu_1 + \mu_2 + \mu_3) = n\), i.e., if the dimension of matrix \( n \) is a factor of \((\mu_1 + \mu_2 + \mu_3)\). To avoid this situation, we have to distinguish two cases: \( n = m(\mu_1 + \mu_2 + \mu_3) \) and \( n \neq m(\mu_1 + \mu_2 + \mu_3) \), \( m \in \mathbb{N} \), i.e.,

\[ \tilde{n} = \begin{cases} n, & n \neq m(\mu_1 + \mu_2 + \mu_3), \quad m \in \mathbb{N}, \\ n - 1, & n = m(\mu_1 + \mu_2 + \mu_3), \quad m \in \mathbb{N}. \end{cases} \]

(14)

In order to avoid the pitfall (b) for a displacement factor \( r \), we take a maximal integer from the set \( F \) which satisfies the inequality

\[-(i - 1)(\mu_1 + \mu_2 + \mu_3) + r\tilde{n} < 0, \quad i = 1 \Longrightarrow r = 0. \]

(15)
REMARK 2. According to (8), the only exception is the design S₆ (direction μ(0,1,1)) where instead of (15) we use the relation

\[ -2(j - 1) + r\bar{n} < 0, \quad j = 1 \implies r = 0. \]

Thus, according to (14) and (15), the compression function which maps \( \bar{P}_{\text{int}}(\mu) \) into \( \hat{P}_{\text{int}}(\mu) \) is defined as

\[ \hat{p}(u, v, w) = \bar{p}(u, v, w) + r\bar{n}\mu. \]  

Let us note that the same compression must be performed in the space \( \bar{P}_{\text{in}}(\mu) \) along the corresponding vectors of data flow. The compression of \( \bar{P}_{\text{in}}(\mu) \) into \( \hat{P}_{\text{in}}(\mu) \) is performed according to the following equations:

\[ \hat{p}_{\gamma}(u, v, w) = \bar{p}_{\gamma}(u, v, w) + r\bar{n}e_{\gamma}^{3}, \quad \gamma \in \{a, b, c\}. \]  

The compression of \( \bar{P}_{\text{int}}(\mu) \) along the line with the direction \( \mu(1,1,1) \) for \( n = 4 \) is illustrated in Figure 2.

3.4. Obtaining the Arrays

By the compression of \( \bar{P}_{\text{int}}(\mu) \), we have transformed graph \( \bar{\Gamma}_{\mu} = (\bar{P}_{\text{int}}(\mu), E) \) into \( \bar{\Gamma}_{\mu} = (\bar{P}_{\text{in}}(\mu), E) \) which is ready for mapping. By mapping graph \( \bar{\Gamma}_{\mu} \) along the corresponding direction \( \mu(\mu_1, \mu_2, \mu_3) \), we can obtain all planar space-time optimal SAs for matrix multiplication. However, in order to provide correct positions of input data items in the projection plane and pipeline processing, we have to expand the space of input computations \( \bar{P}_{\text{in}}(\mu) \) (see [4,5]). This is performed for each direction of the projection \( \mu(\mu_1, \mu_2, \mu_3) \). The new positions of input data elements are defined by

\[ p_{\gamma}(u, v, w) = \bar{p}_{\gamma} - (t(p_{\gamma}) + 1) e_{\gamma}^{3}, \quad \gamma \in \{a, b, c\}. \]  

Thus, we have performed the mapping of \( \bar{P}_{\text{in}}(\mu) \) into \( P_{\text{in}}(\mu) \). The expanding of \( \hat{P}_{\text{in}}(\mu) \) in plane \( k = 1 \), for \( n = 4 \) is pictured in Figure 3. Note that expanding is not applied on \( \bar{P}_{\text{int}}(\mu) \), so we have that \( P_{\text{int}}(\mu) = \bar{P}_{\text{int}}(\mu) \).

The next step is to find the transformation matrix

\[ L(\mu) = \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 \\ \beta_1 & \beta_2 & \beta_3 \end{bmatrix}, \]

which maps a point \( p \in P = P_{\text{in}}(\mu) \cup P_{\text{int}}(\mu) \) into the projection plane. The transformation matrix \( L(\mu) \) maps \( \Gamma_{\mu} = (P_{\text{in}}, E) \) into the projection plane and gives the exact ordering of PEs. This mapping preserves the locality of PEs interconnection. The exact ordering of input data elements in the projection plane is obtained by mapping of \( P_{\text{in}}(\mu) \). Let us note that for a given
direction $\mu(\mu_1, \mu_2, \mu_3)$ the matrix $L(\mu)$ is not uniquely defined. However, all matrices that satisfy the following conditions:
- $L(\mu) \mu = 0$;
- $\alpha_i, \beta_i \in \{-1, 0, 1\}, i = 1, 2, 3$;
- matrix rows are linearly independent;
- $|\alpha_1| + |\alpha_2| + |\alpha_3| > 0$ and $|\beta_1| + |\beta_2| + |\beta_3| > 0$;

can be equally used. Some possible transformation matrices for all allowed directions are given later in the survey of the designs.

The position of each PE in 2-D SA is described by its Cartesian coordinates which are determined according to

$$
\begin{bmatrix}
x \\
y
\end{bmatrix} = L(\mu) \cdot p(u, v, w), \quad p \in P_{\text{int}}(\mu).
$$

Since $L(\mu) \cdot \mu = 0$, instead of $p \in P_{\text{int}}(\mu) = \tilde{P}_{\text{int}}(\mu)$ in (19), according to (16) we can take the corresponding point $\tilde{p} \in \tilde{P}_{\text{int}}(\mu)$. This is possible because the compression is performed along the direction of the projection, so the compression does not affect the image of $\tilde{P}_{\text{int}}(\mu)$. The compression of $\tilde{P}_{\text{int}}(\mu)$ was necessary in order to determine how to compress $\tilde{P}_{\text{in}}(\mu)$.

The positions of input matrix elements in the projection plane are obtained from the following formulae:

$$
\begin{bmatrix}
x \\
y
\end{bmatrix}_\gamma = L(\mu)p_\gamma(u, v, w), \quad \gamma = \{a, b, c\}.
$$
REMARK 3. Instead of (20), the following equivalent formulae can be used:

\[
\begin{bmatrix}
  x \\ y 
\end{bmatrix}_\gamma = L(\mu) \cdot p(\gamma(u, v, w) = L(\mu)p(x)(u, v, w) + r\hat{e}_\gamma^2, \quad \gamma = \{a, b, c\},
\]

(21)

where

\[
p(\gamma(u, v, w) = p(x)(u, v, w) + r\hat{e}_\gamma^3
\]

(22)

and

\[
p(x)(u, v, w) = p_0(u, v, w) - t(p(\gamma(u, v, w)) + 1)\hat{e}_\gamma^2.
\]

(23)

4. SURVEY OF THE OPTIMAL SYSTOLIC DESIGNS

As we have concluded in Section 3.2, there are ten allowable directions of the projection, each of them giving one planar 2-D systolic array for matrix multiplication. In this section, we give a description of each design which contains: direction of the projection, transformation matrix, real spatial positions and positions in the projection plane for processing elements, and input matrix elements. For the sake of illustration, we will present the arrays $S_4$, $S_7$, and $S_8$ obtained by the proposed and standard procedure.

DESIGN $S_1$.
- direction of the projection: $\mu(1, 0, 0)$;
- transformation matrix: $L(\mu) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$;

PE:

\[
p(i, j, k) L(\mu) \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} j \\ k \end{bmatrix},
\]

matrix element $c(i, j, 0)$:

\[
p c(i, j, 2 - i - j) L(\mu) \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} j \\ 2 - i - j \end{bmatrix},
\]

matrix element $a(i, 0, k)$:

\[
p a(i, 2 - i - k, k) L(\mu) \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 2 - i - k \\ k \end{bmatrix},
\]

matrix element $b(0, j, k)$:

\[
p b(2 - j - k, j, k) L(\mu) \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} j \\ k \end{bmatrix},
\]

for $i, j, k = 1, \ldots, n$.

DESIGN $S_2$.
- direction of the projection: $\mu(0, 1, 0)$;
- transformation matrix: $L(\mu) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$;

PE:

\[
p(i, j, k) L(\mu) \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} i \\ k \end{bmatrix},
\]

matrix element $c(i, j, 0)$:

\[
p c(i, j, 2 - i - j) L(\mu) \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} i \\ 2 - i - j \end{bmatrix},
\]

matrix element $a(i, 0, k)$:

\[
p a(i, 2 - i - k, k) L(\mu) \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} i \\ k \end{bmatrix},
\]

matrix element $b(0, j, k)$:

\[
p b(2 - j - k, j, k) L(\mu) \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 2 - j - k \\ k \end{bmatrix},
\]

for $i, j, k = 1, \ldots, n$. 

\[\]
DESIGN $S_3$.
- direction of the projection: $\mu(0,0,1)$;
- transformation matrix: $L(\mu) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$;

$P_E$:

$$p(i,j,k) \xrightarrow{L(\mu)} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} i \\ j \end{bmatrix},$$

matrix element $c(i,j,0)$:

$$p_c(i,j,2-i-j) \xrightarrow{L(\mu)} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} i \\ 2-i-k \end{bmatrix},$$

matrix element $a(i,0,k)$:

$$p_a(i,2-i-k,k) \xrightarrow{L(\mu)} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 2-j-k \\ j \end{bmatrix},$$

matrix element $b(0,j,k)$:

$$p_b(2-j-k,j,k) \xrightarrow{L(\mu)} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} i-j \\ k \end{bmatrix},$$

for $i,j,k = 1, \ldots, n$.

DESIGN $S_4$.
- direction of the projection: $\mu(1,1,0)$,
- transformation matrix: $L(\mu) = \begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & 0 \end{bmatrix}$;

$P_E$:

$$p(i,i+j-1,k) \xrightarrow{L(\mu)} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1-j \\ k \end{bmatrix},$$

matrix element $c(i,i+j-1,0)$:

$$p_c(i,i+j-1,3-2i-j) \xrightarrow{L(\mu)} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1-j \\ 3-2i-j+r \end{bmatrix},$$

matrix element $a(i,0,k)$:

$$p_a(i,2-i-k,k) \xrightarrow{L(\mu)} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 2i+k-2+r \end{bmatrix},$$

matrix element $b(0,i+j-1,k)$:

$$p_b(3-i-j-k,i+j-1,k) \xrightarrow{L(\mu)} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 4-2i-2j-k+r \end{bmatrix},$$

for $i,j,k = 1, \ldots, n$, where

$$\bar{n} = \begin{cases} n, & n \neq 2m, \ m \in \mathbb{N}, \\ n-1, & n = 2m, \ m \in \mathbb{N}, \end{cases}$$

and $r$ is maximal integer from set $F = \{0,1\}$ which satisfies the inequality

$$-2(i-1)+r \bar{n} < 0, \quad \text{if } i = 1, \text{ then } r = 0.$$
**Matrix Multiplication**

Figures 4 and 5 present the array $S_4$, for the case $n = 4$, obtained by the standard and proposed procedure, respectively.

**DESIGN $S_5$.**
- direction of the projection: $\mu(1, 0, 1)$;
- transformation matrix: $L(\mu) = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}$;

**PE:**

\[
p(i, j, i+k-1) \xrightarrow{L(\mu)} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1-k \\ j \end{bmatrix},
\]

matrix element $c(i, j, 0)$:

\[
p_c(i, j, 2-i-j) \xrightarrow{L(\mu)} \begin{bmatrix} x_c \\ y \end{bmatrix} = \begin{bmatrix} 2i+j-2-r\bar{n} \\ j \end{bmatrix},
\]

matrix element $a(i, 0, i+k-1)$:

\[
p_a(i, 3-2i-k, i+k-1) \xrightarrow{L(\mu)} \begin{bmatrix} x_a \\ y_a \end{bmatrix} = \begin{bmatrix} 1-k \\ 3-2i-k+r\bar{n} \end{bmatrix},
\]
matrix element $b(0, j, i + k - 1)$:

$$p_b(3 - i - j - k, j, i + k - 1) L(\mu) \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 4 - 2i - j - 2k + r\bar{n} \\ j \end{bmatrix},$$

for $i, j, k = 1, \ldots, n$, where

$$\bar{n} = \begin{cases} n, & n \neq 2m, \ m \in \mathbb{N}, \\ n - 1, & n = 2m, \ m \in \mathbb{N}, \end{cases}$$

and $r$ is maximal integer from set $F = \{0, 1\}$ which satisfies the inequality

$$-2(i - 1) + r\bar{n} < 0, \quad \text{if } i = 1, \quad \text{then } r = 0.$$

**DESIGN $S_6$.**

- direction of the projection: $\mu(0, 1, 1)$;
- transformation matrix: $L(\mu) = \begin{bmatrix} 0 & 1 & -1 \\ 1 & 0 & 0 \end{bmatrix}$;

PE:

$$p(i, j, k + j - 1) L(\mu) \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 - k \\ i \end{bmatrix},$$
matrix element \( c(i, j, 0) \):

\[
p_c(i, j, 2 - i - j) \overset{L(\mu)}{\rightarrow} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} i + 2j - 2 - r\bar{n} \\ i \end{bmatrix},
\]

matrix element \( a(i, 0, k + j - 1) \):

\[
p_a(i, 3 - i - k - j, k + j - 1) \overset{L(\mu)}{\rightarrow} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 4 - i - 2k - 2j + r\bar{n} \\ i \end{bmatrix},
\]

matrix element \( b(0, j, k + j - 1) \):

\[
p_b(3 - 2j - k, j, k + j - 1) \overset{L(\mu)}{\rightarrow} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 - k \\ 3 - 2j - k + r\bar{n} \end{bmatrix},
\]

for \( i, j, k = 1, \ldots, n \), where

\[
\bar{n} = \begin{cases} n, & n \neq 2m, \ m \in \mathbb{N}, \\ n - 1, & n = 2m, \ m \in \mathbb{N}, \end{cases}
\]

and \( r \) is maximal integer from set \( F = \{0, 1\} \) which satisfies the inequality

\[-2(j - 1) + r\bar{n} < 0, \quad \text{if } j = 1, \quad \text{then } r = 0.\]

**Design \( S_7 \):**

- direction of the projection: \( \mu(1, 1, 1) \);
- transformation matrix: \( L(\mu) = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \end{bmatrix} \);

PE:

\[
p(i, i + j - 1, i + k - 1) \overset{L(\mu)}{\rightarrow} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 - k \\ j - k \end{bmatrix},
\]

matrix element \( c(i, i + j - 1, 0) \):

\[
p_c(i, i + j - 1, 3 - 2i - j) \overset{L(\mu)}{\rightarrow} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 3i + j - 3 - r\bar{n} \\ 3i + 2j - 4 - r\bar{n} \end{bmatrix},
\]

matrix element \( a(i, 0, i + k - 1) \):

\[
p_a(i, 3 - 2i - k, i + k - 1) \overset{L(\mu)}{\rightarrow} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 - k \\ 4 - 3i - 2k + r\bar{n} \end{bmatrix},
\]

matrix element \( b(0, i + j - 1, i + k - 1) \):

\[
p_b(4 - 2i - j - k, i + j - 1, i + k - 1) \overset{L(\mu)}{\rightarrow} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 5 - 3i - j - 2k + r\bar{n} \\ j - k \end{bmatrix},
\]

for \( i, j, k = 1, \ldots, n \), where

\[
\bar{n} = \begin{cases} n, & n \neq 3m, \ m \in \mathbb{N}, \\ n - 1, & n = 3m, \ m \in \mathbb{N}, \end{cases}
\]

and \( r \) is maximal integer from set \( F = \{0, 1, 2\} \) which satisfies the inequality

\[-3(i - 1) + r\bar{n} < 0, \quad \text{if } i = 1, \quad \text{then } r = 0.\]
Figures 6 and 7 present the array $S_T$ for $n = 4$, obtained by the standard and proposed procedure, respectively.

**DESIGN $S_8$.**

- direction of the projection: $\mu(1,1,-1)$;
- transformation matrix: $L(\mu) = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$;

**PE:**

$p(i,i+j-1,k-i+1) \overset{L(\mu)}{\rightarrow} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} k+1 \\ k+j \end{bmatrix},$

matrix element $c(i,i+j-1,0)$:

$p_c(i,i+j-1,n+2-2i-j) \overset{L(\mu)}{\rightarrow} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} n+2-i-j \\ n+1-i \end{bmatrix},$

matrix element $a(i,0,k-i+1)$:

$p_a(i,n-k,k-i+1) \overset{L(\mu)}{\rightarrow} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} k+1 \\ n-i+1 \end{bmatrix},$

matrix element $b(0,i+j-1,k-i+1)$:

$p_b(n+1-i-k,i+j-1,k-i+1) \overset{L(\mu)}{\rightarrow} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} n+2-i-j \\ k+j \end{bmatrix},$

for $k = n,n+1,\ldots,2n-1$ and $i,j = 1,\ldots,n.$
Figures 8 and 9 present the array $S_8$ when $n = 4$, obtained by the standard and proposed procedure, respectively.

**DESIGN $S_8$.**

- direction of the projection: $\mu(1,-1,1)$;
- transformation matrix: $L(\mu) = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}$;

**PE:**

$$p(i,j-i+1,k+i-1) \xrightarrow{L(\mu)} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} j+1 \\ j+k \end{bmatrix},$$

matrix element $c(i,j-i+1,0)$:

$$p_c(i,j-i+1,n-j) \xrightarrow{L(\mu)} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} j+1 \\ n-i+1 \end{bmatrix},$$

matrix element $a(i,0,k+i-1)$:

$$p_a(i,n+2-k-2i,k+i-1) \xrightarrow{L(\mu)} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} n+2-k-i \\ n+1-i \end{bmatrix},$$

matrix element $b(0,j-i+1,k+i-1)$:

$$p_b(n+1-j-k,j-i+1,k+i-1) \xrightarrow{L(\mu)} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} n+2-i-k \\ j+k \end{bmatrix},$$

for $i,k = 1, \ldots, n$ and $j = n, \ldots, 2n-1$. 
Figure 8. The array $S_8$ obtained by the standard procedure ($n = 4$).

Figure 9. The array $S_8$ obtained by the proposed procedure ($n = 4$).
**DESIGN S_{10}**

- Direction of the projection: $\mu(-1, 1, 1)$
- Transformation matrix: $L(\mu) = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$

**PE**

$$p(i, j-i+1, k-i+1) \xrightarrow{L(\mu)} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} j+1 \\ k+1 \end{bmatrix},$$

matrix element $c(i, j-i+1, 0)$:

$$p_c(i, j-i+1, n-j) \xrightarrow{L(\mu)} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} j+1 \\ n+i-j \end{bmatrix},$$

matrix element $a(i, 0, k+i-1)$:

$$p_a(i, n-k, k-i+1) \xrightarrow{L(\mu)} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} n+i-k \\ k+1 \end{bmatrix},$$

matrix element $b(0, j-i+1, k-i+1)$:

$$p_b(n+2i-j-k-1, j-i+1, k-i+1) \xrightarrow{L(\mu)} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} n+i-k \\ n+i-j \end{bmatrix},$$

for $i = 1, \ldots, n$ and $j, k = n, \ldots, 2n-1$.

**5. DISCUSSION**

Table 1 summarizes the characteristics of the systolic arrays $S_i, i = 1, \ldots, 10$, obtained by the proposed procedure and the arrays $S'_i, i = 1, \ldots, 10$, obtained by the standard one. For each SA we give: the number of PEs, time for data input, output, and execution.

<table>
<thead>
<tr>
<th>Class</th>
<th>Design</th>
<th>No. of PE</th>
<th>$t_{in}$</th>
<th>$t_{out}$</th>
<th>$t_{exe}$</th>
<th>$t_{tot}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$S_1, S_2$</td>
<td>$n^2$</td>
<td>$n-1$</td>
<td>0</td>
<td>$3n-2$</td>
<td>$4n-3$</td>
</tr>
<tr>
<td></td>
<td>$S'_1, S'_2$</td>
<td>$n^2$</td>
<td>$n-1$</td>
<td>0</td>
<td>$3n-2$</td>
<td>$4n-3$</td>
</tr>
<tr>
<td></td>
<td>$S_3$</td>
<td>$n^2$</td>
<td>0</td>
<td>$n-1$</td>
<td>$3n-2$</td>
<td>$4n-3$</td>
</tr>
<tr>
<td></td>
<td>$S'_3$</td>
<td>$n^2$</td>
<td>0</td>
<td>$n-1$</td>
<td>$3n-2$</td>
<td>$4n-3$</td>
</tr>
<tr>
<td>2</td>
<td>$S_4-S_6$</td>
<td>$n^2$</td>
<td>$n-1$</td>
<td>0</td>
<td>$3n-2$</td>
<td>$4n-3$</td>
</tr>
<tr>
<td></td>
<td>$S'_4-S'_6$</td>
<td>$2n^2-n$</td>
<td>$n-1$</td>
<td>0</td>
<td>$3n-2$</td>
<td>$4n-3$</td>
</tr>
<tr>
<td></td>
<td>$S_7$</td>
<td>$n^2$</td>
<td>$n-1$</td>
<td>0</td>
<td>$3n-2$</td>
<td>$4n-3$</td>
</tr>
<tr>
<td></td>
<td>$S'_7$</td>
<td>$3n^2-3n+1$</td>
<td>$n-1$</td>
<td>$n-1$</td>
<td>$3n-2$</td>
<td>$5n-4$</td>
</tr>
<tr>
<td>3</td>
<td>$S_8-S_{10}$</td>
<td>$n^2$</td>
<td>0</td>
<td>0</td>
<td>$3n-2$</td>
<td>$3n-2$</td>
</tr>
<tr>
<td></td>
<td>$S'<em>8-S'</em>{10}$</td>
<td>$3n^2-3n+1$</td>
<td>0</td>
<td>0</td>
<td>$3n-2$</td>
<td>$3n-2$</td>
</tr>
</tbody>
</table>

From Table 1, it can be concluded as follows.

- All planar systolic arrays can be grouped into three classes according to interconnection pattern between processing elements (see Figure 10).
- Each of the SAs, obtained by the proposed procedure contains $n^2$ PEs. The arrays $S_1, S_2, S_3$ and $S'_1, S'_2, S'_3$ (Class 1) have the same number of PEs, i.e., $n^2$, since they are obtained by the orthogonal projections. However, the arrays $S'_4, S'_5, S'_6$ (Class 2) have $2n^2-1$ PEs, while $S'_7, S'_{10}$ (Class 3) have $3n^2-3n+1$ PEs, compared with $n^2$ in the SAs obtained by our procedure.
The improvement in space domain was not achieved at the cost of execution time. On the contrary, the total execution time of the arrays $S_i$ and $S'_i$ are the same, except in the case $S_7$, where we have $4n - 3$ time units compared with $5n - 4$ in the $S'_i$.

Using the standard measure, i.e., the product of PE number and required time steps, we conclude that the most convenient systolic arrays for matrix multiplication are $S_8$, $S_9$, and $S_{10}$. This conclusion cannot be derived for the corresponding SAs $S'_8$, $S'_9$, and $S'_{10}$, obtained by the standard procedure.

![Figure 10. Class of systolic arrays according to interconnection patterns.](image)

6. CONCLUSION

This paper is concerned with the problem of synthesizing space-time optimal pure planar systolic arrays for matrix multiplication. This is accomplished by forming a periodical and unconfined computation space which is then used for deriving a separate dependence graph for each allowable direction of the projection. The procedure is an improved version of data dependence method based on linear transformations proposed in [4,6]. By the described procedure, we obtain ten different systolic arrays $S_i$ to $S_{10}$, grouped into three classes according to interconnection pattern between PEs. The systolic arrays designed in accordance with the methodology reported in this paper have following features: $n^2$ processing elements, near-neighbour communications, and execution time of $3n - 2$ time units.

REFERENCES


