FACTA UNIVERSITATIS (NIŠ)
SERIES: ELECTRONICS AND ENERGETICS VOL.13, NO.2, AUGUST 2000, 157–165

MATRIX MULTIPLICATION ON NON-PLANAR SYSTOLIC ARRAYS

Teufik I. Tokić, Emina I. Milovanović, Natalija M. Novaković, Igor Ž. Milovanović and Mile K. Stojčev

Abstract. A modification of standard design procedure for mapping nested loop algorithms into systolic arrays is described in this article. This modification enables us to obtain non–planar systolic arrays for matrix multiplication with optimal number of processing elements for a given problem size. The modification is based on composition of two linear mappings.

Key words: Systolic arrays, Matrix multiplication.

1. Introduction

There are totally nineteen SAs that can be obtained by the systematic design methodology for mapping algorithms into systolic architecture [5,8]. SAs obtained by this methodology can be grouped into four classes according to the interconnection pattern between the processing elements (PE): mesh–connected (3 SAs – class_1), orthogonal (3 SAs – class_2), hexagonal (4 SAs – class_3), and non–planar (9 SAs – class_4). Most of the papers concern with the first three classes [1–5] since non–planar SAs obtained by the standard systematic design methodology have great number of PEs. For example, if the dimensions of matrices which are multiplied are $n \times n$, the arrays from class_4 have $4n^2 - 5n + 2$ PEs [6, 8], compared with n^2 PEs in the class_1 arrays.

In this paper we propose a modification of the standard systematic design procedure which enables us to obtain the class_4 SAs with optimal number of PEs, i.e. n^2. The modification is based on the composition of

Manuscript received April 22, 2000. A version of this paper was presented at the fourth IEEE Conference on Telecommunications in Modern Satellite, Cables and Broadcasting Services, TELSIKS'99, October 1999, Niš, Yugoslavia.

The author are with School of Electronic Engineering, P.O.Box 73, Beogradska 14, 18000 Niš, Serbia.

two linear mappings. The first one accommodates the index space of the matrix multiplication algorithm to the direction projection, μ. The second, maps the accommodated index space onto SA architecture. Further, we use different SAs performance measures to compare the obtained SAs with the results found in [8].

2. Standard Synthesis Procedure

Suppose $A = (a_{ik})$ and $B = (b_{kj})$ are two $n \times n$ matrices. A nested loop algorithm for computing their product $C = A * B$ is as follows

```
Algorithm_1
 for $k := 1$ to $n$ do
  for $j := 1$ to $n$ do
   for $i := 1$ to $n$ 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

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

The computational structure of Algorithm_1 is characterized by the index space, P_{int}, where the data are used or computed, i.e.

$$P_{int} = \{(i,j,k) | 1 \leq i,j,k \leq n\}, \tag{2.1}$$

and a dependency matrix which consists of a set of constant dependency vectors, $D = [\vec{e}_b^{\,3}\ \vec{e}_a^{\,3}\ \vec{e}_c^{\,3}]$, each of them representing a data dependency corresponding to one of three variables (b, a, c, respectively). Matrix D for Algorithm_1 is given by

$$D = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.2}$$

Given a dependency matrix D, and index space P_{int}, all allowable projection directions $\vec{\mu} = [\mu_1\, \mu_2\, \mu_3]^T$, that give all planar and non–planar systolic arrays for matrix multiplication can be derived. In this paper we are concerned with non–planar arrays, only. We call them the "X–arrays" because of the topology of interconnection pattern between the PEs. There are totally nine projection directions that give the arrays with this topology.

They are $\vec{\mu}=[1\,1\,2]^T$, $\vec{\mu}=[1\,2\,1]^T$, $\vec{\mu}=[2\,1\,1]^T$, $\vec{\mu}=[1\,-1\,2]^T$, $\vec{\mu}=[-1\,1\,2]^T$, $\vec{\mu}=[1\,2\;-1]^T$, $\vec{\mu}=[2\,1\;-1]^T$, $\vec{\mu}=[-1\,2\,1]^T$, and $\vec{\mu}=[2\;-1\,1]^T$. Since the design methodology is similar for all directions, without lost of generality we will concern the array obtained by the direction $\vec{\mu}=[2\,1\;-1]^T$. The obtained results will be compared with one given in [8].

Each allowable projection direction is associated with the corresponding space–time transformation matrix T which maps a computational structure of the algorithm (D, P_{int}) into a systolic implementation. Matrix T which corresponds to the direction $\vec{\mu}=[2\,1\;-1]^T$, is of the form

$$T=\begin{bmatrix}\vec{\Pi}\\ S\end{bmatrix}=\begin{bmatrix}1 & 1 & 1\\ -- & -- & --\\ 0 & 1 & 1\\ 1 & -1 & 1\end{bmatrix} \tag{2.3}$$

where $\vec{\Pi}=[1\,1\,1]$ determines time scheduling, and

$$S=\begin{bmatrix}0 & 1 & 1\\ 1 & -1 & 1\end{bmatrix}$$

is a space transformation which maps P_{int} into systolic array.

According to the standard synthesis procedure [8] the (x, y) positions of the PEs in the SA are determined by

$$\begin{bmatrix}x\\ y\end{bmatrix}=S\cdot\begin{bmatrix}i\\ j\\ k\end{bmatrix}=\begin{bmatrix}0 & 1 & 1\\ 1 & -1 & 1\end{bmatrix}\cdot\begin{bmatrix}i\\ j\\ k\end{bmatrix}=\begin{bmatrix}j+k\\ i-j+k\end{bmatrix} \tag{2.4}$$

for $1\le i,j,k\le n$. The communication links between the PEs are implemented along the projections of data dependency vectors, i.e.

$$\Delta_S=S\cdot D=[\vec{e}_b^{\,2}\;\vec{e}_a^{\,2}\;\vec{e}_c^{\,2}]=\begin{bmatrix}0 & 1 & 1\\ 1 & -1 & 1\end{bmatrix}. \tag{2.5}$$

The space of initial computations of Algorithm_1, $P_{in}=\{P_{in}(a),\ P_{in}(b),\ P_{in}(c)\}$, is defined as $P_{in}(a)=\{(i,0,k)|1\le i,k\le n\}$, $P_{in}(b)=\{(0,j,k)|\ 1\le j,k\le n\}$, $P_{in}(c)=\{(i,j,0)|1\le i,j\le n\}$

In order to obtain the correct time ordering of input data items in the 2D SA, the initial computation space has to be reordered. The reordering is performed as follows. Let $\vec{p}_\gamma$ be a position vector of data item γ, $\gamma \in \{a, b, c\}$ in the space P_{in}. The new position in reordered space P^*_{in} is obtained from

$$\vec{p}^{\,*}_\gamma = \vec{p}_\gamma - (t(\vec{p}_\gamma) + 1)\vec{e}^{\,3}_\gamma, \quad \gamma \in \{a, b, c\} \tag{2.6}$$

where $t(p) = t(i, j, k) = i + j + k - 3$ is a timing function which defines a temporal distribution of the computation. The initial (x, y) positions of input data items in the projection plane are obtained according to

$$\gamma(i,\, j,\, k) \longmapsto \begin{bmatrix} x \\ y \end{bmatrix}_\gamma = S \cdot \vec{p}^{\,*}_\gamma, \quad \gamma \in \{a, b, c\}.$$

Data distribution at the beginning of the computation in this array is depicted in Fig.1 for $n = 3$. Space–time parameters, which include total execution time, T_{tot}, number of PEs in the SA, geometric and chip area, speed–up, efficiency, AT and AT^2 measures, are given in Table 1.

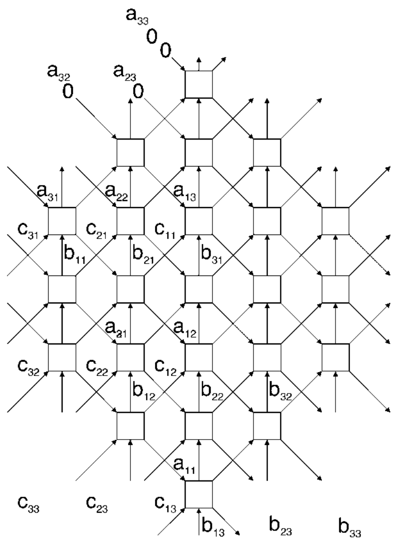

Fig. 1. Data distribution at the beginning of the computation in the non-planar array synthesized by the standard design procedure, for $n = 3$

3. Modification of the Synthesis Procedure

In order to obtain non–planar SAs with better space–time parameters, we involve a modification of the standard synthesis procedure. The crucial novelty in our approach is that we do not perform mapping

$$(D, P_{int}) \xmapsto{T} (\Delta, \bar{P}_{int}) \tag{3.1}$$

directly. Instead, we substitute (3.1) with the following two mappings

$$P_{int} \xmapsto{H} P^*_{int}, \quad \text{and } (D, P^*_{int}) \xmapsto{T} (\Delta, \bar{P}_{int}). \tag{3.2}$$

where H is transformation that accommodates index space P_{int} to the direction $\vec{\mu} = [2\,1\;-1]^T$. Namely, the features of the operations in Algorithm_1 enable that the computation of C elements can be performed over arbitrary permutation of sequence $(k_1 \cdots k_n)$ of index variable $k = 1, \dots, n$. The mapping $H = (F, G)$ is defined as

$$F = \begin{bmatrix} 1 & 2 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix}, \qquad \text{and } G = \begin{bmatrix} -2 \\ 0 \\ n \end{bmatrix} \tag{3.3}$$

The mapping of index set $\{(i, j, k)\}$ into new index set $\{(u, v, w)\}$ using transformation H is defined by

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = F \begin{bmatrix} i \\ j \\ k \end{bmatrix} + G = \begin{bmatrix} i + 2j - 2 \\ j \\ k - j + n \end{bmatrix}, \tag{3.4}$$

for each $[i\,j\,k]^T \in P_{int}$, $1 \le i, j, k \le n$.

Now we will show that transformation H really accommodate P_{int} to the direction $\vec{\mu} = [2\,1\;-1]^T$. Let $[i\,j_1\,k]^T$ and $[i\,j_2\,k]^T$, $j_1 \neq j_2$, be two points from the space P_{int}. Their images in P^*_{int} are $[u_1\,v_1\,w_1]^T$ and $[u_2\,v_2\,w_2]^T$, respectively. Then according to (3.4) we have that

$$\begin{bmatrix} u_1 \\ v_1 \\ w_1 \end{bmatrix} - \begin{bmatrix} u_2 \\ v_2 \\ w_2 \end{bmatrix} = (j_1 - j_2) \begin{bmatrix} 2 \\ 1 \\ -1 \end{bmatrix} = (j_1 - j_2) \cdot \vec{\mu}.$$

This means that for some fixed i and k there are exactly n, $(j = 1, \dots, n)$, different index points from P_{int} lying on the line parallel to direction $\vec{\mu} = [2\,1\;-1]^T$. Since $1 \le i, k \le n$, there are exactly n^2 such lines. Consequently, when P^*_{int} is mapped by T into $\bar{P}_{int}$, this set will have n^2 elements, i.e. the corresponding SA will contain n^2 PEs. The (x, y) positions of the PEs in the SA are obtained according to composite mapping $(S \cdot H)$ as follows

$$p(i,j,k) \xmapsto{H} p^*(i+2j-2, j, k-j+n) \xmapsto{S} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} k+n \\ k+i+n-2 \end{bmatrix}. \quad (3.5)$$

Communication links between the PEs are of near-neighbor type implemented along the directions (2.5).

Having obtained PE positions, the next step is to find out the distribution of A, B and C elements at the beginning of the computation in the systolic array. Note that mapping of P_{int} into P^*_{int} requires that the initial computation space P_{in} has to be mapped, also. The new initial space $\hat{P}_{in}$, is defined by $\hat{P}_{in}(a) = \{(i+2j-2, 0, k-j+n) | 1 \le i,j,k \le n\}$, $\hat{P}_{in}(b) = \{(0, j, k-j+n) | 1 \le j,k \le n\}$, $\hat{P}_{in}(c) = \{(i+2j-2, j, 0) | 1 \le i,j \le n\}$ with the periodicity of A, B, and C data items defined by $a(i,j,k) \equiv a(i+n,j,k) \equiv a(i,j,k+n) \equiv a_{ik}$, $b(i,j,k) \equiv b(i,j,k+n) \equiv b_{kj}$, $c(i,j,k) \equiv c(i+n,j,k) \equiv c(i,j,k+n)$. Now, according to the timing function and equation (2.6), this space is expanded as follows

$$\begin{aligned} P^*_{in}(a) &= \{(i+2j-2, 3-i-j-k, k-j+n) | 1 \le i,j,k \le n\} \\ P^*_{in}(b) &= \{(1-k, j, k-j+n) | 1 \le j,k \le n\} \\ P^*_{in}(c) &= \{(i+2j-2, j, n-i-3j+3) | 1 \le i,j \le n\} \end{aligned} .$$

If we now apply mapping $S : P^*_{in} \longmapsto \bar{P}_{in}$, we would obtain the initial ordering of input elements in the SA according to (3.5). This array has n^2 processing elements, but long execution time. Therefore we have to perform time optimization. Optimization in time domain is performed as follows. Timing function of the array obtained according to (3.5) is

$$t(i+2j-2, j, k-j+n) = i+2j+k-4. \quad (3.6)$$

For some fixed i and k and for $j := j$ and $j := j+1$ according to (3.6) we have that

$$\Delta t = t(i+2j, j+1, k-j+n-1) - t(i+2j-2, j, k-j+n) = 2. \quad (3.7)$$

The above equality implies that data items enter the array in every second time instance. To optimize time parameter of the array we have to achieve that data enter the array in consecutive time moments. Therefore we have to reorder data distribution. We call this reordering a compression. The compression is performed as translation of data item along the direction of data flow $\vec{e}_\gamma^{\,2}$, $\gamma \in \{a, b, c\}$ over index variable j. There are two things that we should take care of during this compression:

i) two data items must not overlap

ii) no data item can enter the array before the one indexed by $j = 1$, $(i, k$ arbitrary).

To avoid this traps we perform the following analysis. Let $(i,\, j_1,\, k)$ and $(i,\, j_2,\, k)$, $j_1 \neq j_2$ be two arbitrary points. Suppose that after the translation of data item indexed by $(i,\, j_2,\, k)$ for factor n (n is dimension of matrix) it coincides now with data item indexed by $(i,\, j_1,\, k)$. In that case the following is valid

$$t(i+j_2-2, j_2, k-j_2+n) - n = t(i+2j_1-2, j_1, k-j_1+n), \text{ i.e. } n = 2(j_2 - j_1).$$

This means that overlapping of data items can occur if n is even. Therefore we have to make a difference when n is even and when n is odd. For the sake of simplicity we introduce the following notation

$$\bar{n} = \begin{cases} n, & \text{when } n \text{ is odd} \\ n-1, & \text{when } n \text{ is even} \end{cases}.$$

Since in (3.7) $\Delta t = 2$, data items are translated for a factor $r\bar{n}$, where $r \in \{0, 1\}$. The r is determined from the condition ii), i.e. from the inequality

$$t(i+2j-2, j, k-j+n) - r\bar{n} > t(i, 1, k+n-1).$$

Thus we obtain that r is the greatest integer from the set $\{0, 1\}$ that satisfies the inequality

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

Now, the (x, y) positions of input data items at the beginning of the computation are given by

$$\begin{aligned} a(i+2j-2, 0, k-j+n) &\to \begin{bmatrix} x \\ y \end{bmatrix}_a = \begin{bmatrix} 3+n-i-2j \\ n+2i+2j+2k-5 \end{bmatrix} + r\bar{n} \begin{bmatrix} 1 \\ -1 \end{bmatrix} \\ b(0, j, k-j+n) &\to \begin{bmatrix} x \\ y \end{bmatrix}_b = \begin{bmatrix} k+n \\ n-2j+1 \end{bmatrix} + r\bar{n} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \\ c(i+2j-2, j, 0) &\to \begin{bmatrix} x \\ y \end{bmatrix}_c = \begin{bmatrix} n-i-2j+3 \\ n-2j+1 \end{bmatrix} + r\bar{n} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \end{aligned} \tag{3.8}$$

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

Equations (3.5) and (3.8) give new non–planar systolic array that implements Algorithm_1. Data distribution at the beginning of the computation in this array is given in Fig.2, for $n = 3$.

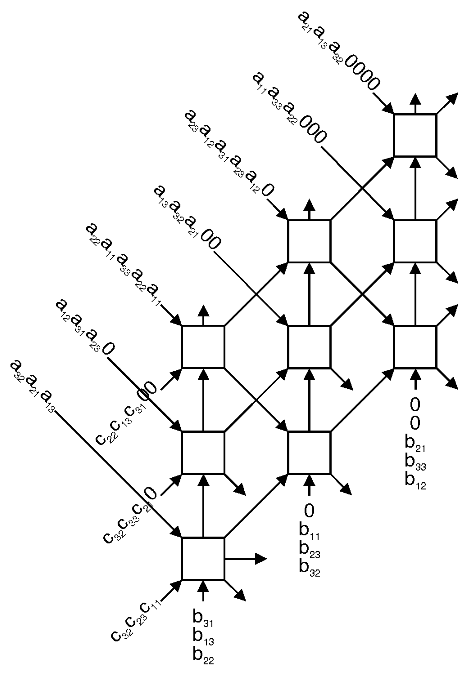

Fig.2. Data distribution at the beginning of the computation in the non-planar array synthesized by the described procedure, for $n = 3$

4. Discussion

A comparative survey of the space-time parameters of the array obtained by the modified design procedure and the one obtained in [8] is given in Table 1.

According to Table 1 we conclude the following. The total execution time, T_{tot}, of the Algorithm_1 is identical for both arrays, but all other parameters of the array obtained by the modified synthesis procedure are substantionally better than that of the array obtained in [8]. For example, the number of PEs in the array synthesized in this paper is almost four times less than the number of PEs in the array given in [8]. Let us note that all other non–planar SAs can be synthesized using the proposed procedure.

Table 1.

Performance measure	SA obtained in [8]	SA obtained in this paper
No of PEs	$4n^2 - 5n + 2$	n^2
T_{tot}	$3n - 2$	$3n - 2$
Geometric area	$4(n-1)^2$	$(n-1)^2$
Chip area	$(2n-1)(3n-2)$	$n(2n-1)$
Speedup*	$\frac{n^3}{3n-2}$	$\frac{n^3}{3n-2}$
Efficiency	$\frac{n^3}{(3n-2)(4n^2-5n+2)}$	$\frac{n}{3n-2}$
AT	$(3n-2)(4n^2-5n+2) = O(12n^3)$	$(3n-2)n^2 = O(3n^3)$
AT^2	$(4n^2-5n+2)(3n-2)^2 = O(36n^4)$	$n^2(3n-2)^2 = O(9n^4)$

*Execution time of matrix multiplication algorithm on uniprocessor system is taken to be n^3 time units. Duration of add–multiply operation is taken as a time unit

REFERENCES

1. J. A. B. Fortes, K. S. Fu, B. W. Wah: *Systematic Design Approaches for Algorithmically Specified Systolic Arrays.*, (V. Milutinović, ed), Computer Architectures, Elsevier Ltd., New York, 1988, 454-494.
2. G.-J. Li, B.W. Wah: *The Design of Optimal Systolic Arrays.*, IEEE Trans. Comput., **34 (1)** (1985), 66-77.
3. 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., **3 (6)** (1997), 17-35.
4. D.I. Moldovan: *Parallel Processing: From Applications to Systems.*, Morgan Kaufman Publishers, San Mateo, 1993.
5. S.G. Sedukhin: *The Designing and Analysis of Systolic Algorithms and Structures.*, Programming, **2** (1990), 20-40. (In Russian)
6. H. V. Jagadish, S. K. Rao, T. Kailath: *Array architectures for iterative algorithms.*, Proc. IEEE, Vol.75, 9 (1989), 149-155.
7. H. V. Jagadish, T. Kailath: *A Family of New Efficient Arrays for Matrix Multiplication.*, IEEE Trans. Comput. **38 (1)** (1989), 149-155.
8. C. R. Wan, D. J. Evans: *Nineteen Ways of Systolic Matrix Multiplication.*, Intern. J. Computer Math., Vol. 68 (1998), 39-69.