FILOMAT 17 (2003), 135–141

MATRIX MULTIPLICATION ON BIDIRECTIONAL LINEAR SYSTOLIC ARRAYS

I. Ž. MILOVANOVIĆ, E. I. MILOVANOVIĆ, B. M. RANDJELOVIĆ, AND I. Č. JOVANOVIĆ

Abstract. This paper addresses the problem of rectangular matrix multiplication on bidirectional linear systolic arrays (SAs). We analyze all bidirectional linear SAs in terms of efficiency. We conclude that the efficiency depends on the relation between the loop boundaries in the systolic algorithm (i.e. matrix dimensions). We point out which SA is the best choice depending on the relation between matrix dimensions. We have designed bidirectional linear systolic arrays suitable for rectangular matrix multiplication.

1. Introduction

Matrix multiplication plays a central role in numerical linear algebra, since one has to compute this product at 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 their 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

1991 *Mathematics Subject Classification.* Primary: 68M07, Secondary: 68Q35 .
Key words and phrases. Matrix multiplication, systolic arrays, efficiency.

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 the 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 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 the implementation of matrix multiplication.

2. Background

Let $A = (a_{ik})_{N_1 \times N_3}$ and $B = (b_{kj})_{N_3 \times N_2}$ be two rectangular matrices. For the systolic implementation of their product, $C = A \cdot B$, 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 $a(i, 0, k) \equiv a_{ik}$, $b(0, j, k) \equiv b_{kj}$, $c(i, j, 0) \equiv 0$, $c_{ij} \equiv c(i, j, N_3)$.

The matrix multiplication problem, and consequently, Algorithm_1, is a three-dimensional one. According to Algorithm_1, orthogonal two-dimensional (2D) SAs can be synthesized (see, for example [3, 4, 9, 10, 15, 17]). Orthogonal 2D SAs are obtained by the following projection direction vectors $\vec{\mu} = [1\,0\,1]^T$, $\vec{\mu} = [0\,1\,1]^T$ and $\vec{\mu} = [1\,1\,0]^T$. In order to obtain orthogonal SAs with an optimal number of processing elements (PEs) for a given problem size, Algorithm_1 has to be adjusted to the direction projection vectors either on the index variable i or j, depending on the relation between N_1 and N_2 (i.e $N_1 > N_2$ or $N_1 < N_2$) [1, 2, 16]. If we put $N_1 = 1$ (or $N_2 = 1$) 2D orthogonal SAs degrade to 1D bidirectional SAs suitable for the implementation of matrix-vector products (see, for example [10, 11, 12, 13]. This is valid only for the direction projections $\vec{\mu} = [0\,1\,1]^T$ and $\vec{\mu} = [1\,0\,1]^T$. This fact can be used to compute matrix products on 1D bidirectional SAs iteratively, by repeating the procedure N_1 (i.e. N_2) times. Details concerning the synthesis of 1D bidirectional SAs for matrix-vector multiplication can

be found in [10, 11, 12, 13]. Here we will give some performance measures of bidirectional 1D SAs when used for computing matrix products.

Denote by SA1 and SA2 bidirectional 1D arrays synthesized for the case $i = 1$ and $\vec{\mu} = [0\,1\,1]^T$ and $j = 1$ and $\vec{\mu} = [1\,0\,1]^T$, respectively. Both of these SAs are optimal with respect to the number of PEs for a given problem size and consist of $\Omega = N_3$ PEs. The total computation time of SA1 in the case of matrix multiplication is $T_{tot}^{(1)} = N_1(N_2 + 2N_3 - 2)$, whereas for SA2 it is $T_{tot}^{(2)} = N_2(N_1 + 2N_3 - 2)$. Consequently, we conclude that when $N_2 > N_1$ the array SA1 is the better choice and SA2 otherwise. The data flow in SA2 for the case $N_1 = 3$, $N_2 = 2$ and $N_3 = 5$ is depicted in Fig. 1. The efficiency of the arrays SA1 and SA2 are

$$E^{(1)} = \frac{N_2}{N_2 + 2N_3 - 2} \quad \text{and} \quad E^{(2)} = \frac{N_1}{N_1 + 2N_3 - 2}, \tag{2.1}$$

respectively. According to (2.1) one can conclude that when $N_3 >> N_2$ (or $N_3 >> N_1$) the efficiency of the array SA1 (i.e. SA2) is very low despite the fact that these arrays are optimal with respect to the number of PEs. Having this in mind the goal of this paper is to synthesize 1D bidirectional SAs which are efficient for a computing matrix product in the case $N_3 > N_1, N_2$.

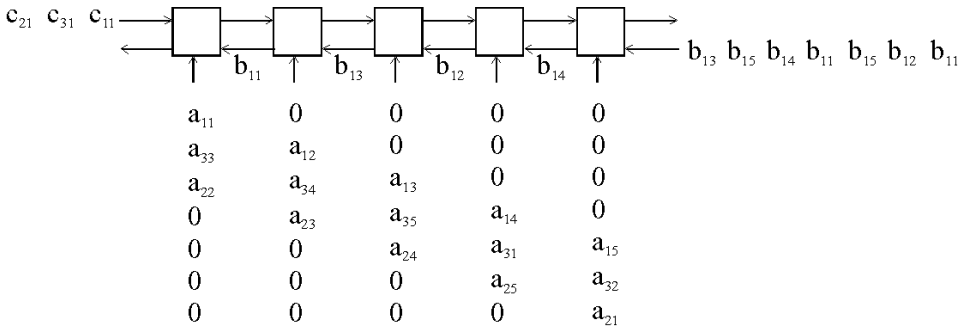

FIGURE 1. Data flow in the array SA_2, for $N_1 = 3, N_2 = 2, N_3 = 5$

Before we proceed with the synthesis of these arrays we will point out some facts. Namely, in the case of the arrays SA1 and SA2, the pipelining is achieved across one of the input matrices and the resulting matrix. Since the index variable k in Algorithm_1 is an iterative one, it is not subject to any transformation, usually. Therefore we did not discuss the case $k = 1$. But if we want to design efficient arrays when $N_3 > N_1, N_2$, we have to abandon this rule. This is the subject of the next section.

3. A new class of 1D bidirectional systolic arrays

Suppose that $N_3 = 1$ in Algorithm_1 (i.e. $k = 1$). In this case Algorithm_1 represents an algorithm for computing the outer product of two vectors, i.e. the product of the first column-vector of a matrix A and first row-vector of a matrix B. The result of this product are the first iterations, $C^{(1)} = (c_{ij}^{(1)})$, of the matrix $C = A \cdot B$. In other words, the computation of $C = A \cdot B$ can be performed according to

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

where $\vec{A}_{.k}$ is the k-th column-vector of matrix A, and $\vec{B}_{k.}$ is the k-th row-vector of matrix B, for $k = 1, 2, \ldots, N_3$.

In order to compute a matrix product on a 1D bidirectional SA we have to synthesize the array which computes $C^{(1)} = \vec{A}_{.1} \vec{B}_{1.}$ and then repeat the computation N_3 times. This array can be synthesized according to Algorithm_1, by substituting $k = 1$, and the direction $\vec{\mu} = [1\,1\,0]^T$. The obtained 1D SA represents a degraded 2D orthogonal SA. To obtain an SA with an optimal number of PEs we have to adjust Algorithm_1 to the direction $\vec{\mu} = [1\,1\,0]^T$.The adjustment can be performed over the index variables i and j. The adjustment is performed by skewing one index variable with respect to the other one.

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

Algorithm_2
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 $a(i, j+N_2, 1) \equiv a(i, j, 1) \equiv a(i, 0, 1) \equiv a_{i1}$, $b(i, j+N_2, 1) \equiv b(i, j, 1) \equiv b(0, j, 1) \equiv b_{1j}$, $c(i, j, 0) \equiv 0$, $c(i, j+N_2, 1) \equiv c(i, j, 1) \equiv c_{ij}^{(1)}$, for $i = 1, 2, \ldots N_1$ and $j = 1, 2, \ldots N_2$.

The bidirectional SA synthesized according to Algorithm_2, denoted as SA3, has the following performances: number of PEs, $\Omega = N_2$, total execution time, $T_{tot}^{(3)} = N_3(N_1 + 2N_2 - 3)$, and the efficiency

$$E^{(3)} = \frac{N_1}{N_1 + 2N_2 - 2}. \tag{3.1}$$

As one can see the efficiency of SA3 does not depend on N_3. Consequently it is more efficient than both SA1 and SA2 when $N_3 > \max\{N_1, N_2\}$. The

data flow in SA3 for the case $N_1 = 3$, $N_2 = 2$ and $N_3 = 5$ is depicted in Fig. 2. The elements of the input matrices, A and B, are pipelined through the array.

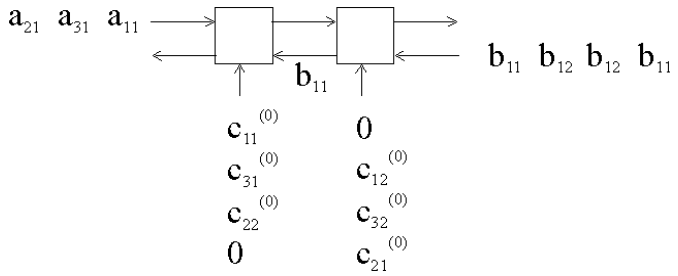

FIGURE 2. Data flow in the array SA_3, for $N_1 = 3, N_2 = 2, N_3 = 5$

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

Algorithm_3

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 $a(i + N_1, j, 1) \equiv a(i, j, 1) \equiv a(i, 0, 1) \equiv a_{i1}$, $b(i + N_1, j, 1) \equiv b(i, j, 1) \equiv b(0, j, 1) \equiv b_{1j}$, $c(i, j, 0) \equiv 0$, $c(i + N_1, j, 1) \equiv c(i, j, 1) \equiv c_{ij}^{(1)}$, for $i = 1, 2, \ldots N_1$ and $j = 1, 2, \ldots N_2$.

The corresponding bidirectional SA, denoted as SA4, which implements Algorithm_3 has the following performances: number of PEs, $\Omega = N_1$, total computation time to find the matrix product, $T_{tot}^{(4)} = N_3(N_2 + 2N_1 - 2)$, and the efficiency

$$E^{(4)} = \frac{N_2}{N_2 + 2N_1 - 2}. \tag{3.2}$$

As one can see neither the efficiency of SA4 depends on N_3 and it is more efficient than SA1 and SA2 in the case $N_3 > \max\{N_1, N_2\}$.

4. Conclusion

In this paper we have designed two new bidirectional linear systolic arrays, denoted as SA3 and SA4, which can be used to compute matrix products. This completes the class of bidirectional linear SAs. We have compared the

arrays SA1, SA2, SA3 and SA4 in term of efficiency. We conclude that the efficiency depends on the relation between the loop boundaries N_1, N_2 and N_3. If $N_1 > N_3 > N_2$ or $N_3 > N_1 > N_2$ the best choice in term of efficiency is the array SA3. If $N_2 > N_3 > N_1$ or $N_3 > N_2 N_1$, the best choice is the array SA4. If $N_1 > N_2 > N_3$, the array SA2 is the best choice. Finally, if $N_2 > N_1 > N_3$ the best choice is SA1.

References

[1] M. Bekakos, E. Milovanović, N. Stojanović, T. Tokić, I. Milovanović, I. Milentijević, Transformation matrices for systolic array synthesis, *J. Electrotech. Math.,* **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, 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.

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

[5] H. T. Kung, Why systolic architectures?, *Computer*, **15**, (1982), 37–46.

[6] 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.

[7] S. Y. Kung, *VLSI array processors*, Prentice Hall, New Jersey, 1988.

[8] P. Lee, Z. M. Kedem, Synthesizing linear array algorithms from nested loop algorithms, *IEEE Trans. Comput.*, **37(12)**, (1988), 1578–1598.

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

[10] 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.*, **33(6)**, (1997), 17–35.

[11] E. I. Milovanović, M. B. Tošić, I. Ž. Milovanović, I. Z. Milentijević, Designing processor-time optimal systolic arrays for matrix-vector multiplication, *J. Electrotechn. Math.*, **1**, (1998), 7–19.

[12] 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.*, **32(2)**, (1996), 21–31.

[13] 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.*, **4**, (1999), 35–40.

[14] I. V. Ramankrishnan, D. S. Fussell, A. Silberschatz, Mapping homogenous graphs on linear arrays, *IEEE Trans. Comput.*, **35(3)**, (1986), 189–209.

[15] 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).

[16] 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.

[17] C. R. Wan, D. J. Evans, Nineteen ways of systolic matrix multiplication, *Inter. J. Comput. Math.*, **68**, (1998), 39–69.

[18] J. Xue, Closed-form mapping conditions for the synthesis of linear processor arrays, *J. VLSI Signal Processing*, **10(2)**, (1995), 181–189.

[19] J. Xue, C. Lengauer, The synthesis of control signals for one-dimensional systolic arrays, *Integration - The VLSI Journal*, **14(1)**, (1992), 1–32.

(I.Milovanović, E. Milovanović, B. Randjelović) Faculty of Electronic Engineering, Beogradska 14a, P.O. Box 73, 18000 Niš,Serbia

(I. Jovanović) Faculty of Scienceand Mathematics, Višegradska 33, 18000 Niš, Serbia

E-mail address, Igor Milovanović: igor@elfak.ni.ac.yu