SCIENTIFIC PUBLICATIONS OF THE STATE UNIVERSITY OF NOVI PAZAR
SER. A: APPL. MATH. INFORM. AND MECH. vol. 3, 2 (2011), 153-159.

Implementation of Optimal Investment Problem on a Linear Systolic Array

T. Z. Mirković, D. Ć. Dolićanin, I. Ž. Milovanović, E. I. Milovanović

Abstract: One of the main problems in operational research is the problem of optimal investments. This paper describes a procedure for synthesis a linear systolic array that implements the algorithm for solvin a problem of optimal investments. The performances of the obtained array, including execution time, number of processing elements, speed-up and efficiency are then discussed.

Keywords: Systolic arrays, optimal investments problem

1 Introduction

One of the main problems in operational research is the problem of optimal investments. The goal is to determine the distribution of a budget of n units of money to $m, m \leq n$, investment programs and achieve maximum benefit. For each investment program the benefit is known in advance if quantity of $i, 0 \leq i \leq n$, units of money is invested in it. The benefits are defined by the corresponding rectangular matrix $M = (a_{ri})_{m \times n+1}$, of nonnegative numbers. The element a_{ri} denotes the benefit achieved by the r-th investment program if $i-1$ units of money is invested in it. Solving problem of optimal investment can be expressed as a transformation of matrix $M = (a_{ri})_{m \times n+1}$ into matrix $\overline{M} = (\overline{a}_{ri})_{m \times n+1}$, whose last row represents the solution of a given problem. This transformation can be described as [1]:

$$\overline{a}_{r,n+2-i} = \max_{k}\{a_{r,k}, \overline{a}_{r-1,n+3-k-1}\} \tag{1}$$

for each $i = 1,2,\ldots,n+1$, $k = 1,2,\ldots,n+2-i$ and $r = 2,3,\ldots,m$. The initial values are $\overline{a}_{1i} \equiv a_{1i}$, $i = 1,2,\ldots,n+1$.

Computational tasks can be conceptually classified into two families: compute-bound computations and I/O-bound computations [2]. For example, matrix multiplication represents compute bound computation. On the other hand, adding two matrices is I/O-bound

Manuscript received December 17, 2010 ; accepted March 29, 2011.

T. Z. Mirković, D. Ć. Dolićanin are with the State University of Novi Pazar, Novi Pazar, Serbia; I. Ž. Milovanović, E. I. Milovanović are with the Faculty of Electronic Engineering, University of Niš, Niš, Serbia;.

task. A problem of optimal investments falls into the category of compute-bound tasks. Speeding-up a compute-bound computations can often be accomplished in a relatively simple and inexpensive manner, that is by the systolic approach, without increasing I/O requirements. Systolic arrays (SA) are high-performance, special purpose architectures typically used to meet specific application requirements or to off-load computations that are especially taxing to general purpose computers. The major features of adopting SA for special purpose processing architectures are: simple and regular design, concurrency, near-neighbour communication and balancing computations with the I/O. 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.

2 Synthesis of the basic systolic array

The first step in systolic array synthesis is constructing a systolic algorithm for a given problem [3]-[6]. Therefore we have to construct The systolic algorithm for the computations defined by (1).

The computations in (1) are identical with respect to r, and are repeated m-1 times. Therefore we will describe the synthesis of a bidirectional SA for some fixed r. Without affecting the generality we take $r = 2$. We will denote this array as the basic one. Having this in mind and in order to simplify the denotation, we introduce the following indication

$$\bar{a}_{2,n+2-i} = c_{n+2-i}, \quad \bar{a}_{1,n+3-i-k} = b_{n+3-i-k}, \quad a_{2,k} = a_k,$$

for each $i = 1,2,\ldots,n+1$, and $k = 1,2,\ldots,n+2-i$. Now, we can rewrite (1), for r=2, in the form of recurrence relation

$$c^{(k)}_{n+2-i} = \max_k\{c^{(k-1)}_{n+2-i}, a_k + b_{n+3-i-k}\} \tag{2}$$

where $c^{(0)}_{n+2-i} \equiv 0$, for each each $i = 1,2,...,n+1$.

The corresponding systolic algorithm has the following form

Algorithm_1

for $k := 1$ **to** $n+1$ **do**
for $i := 1$ **to** k **do**
$a(i,1,k) := a(i,0,k)$**;**
$b(i,1,k) := b(i-1,1,k)$**;**
$c(i,1,k) := \max\{c(i,1,k-1), a(i,1,k) + b(i,1,k)\}$
endfor$\{i,k\}$.

The computational structure of the above algorithm is determined by the inner computation space

$$P_{int} = \{(i,1,k) \mid 1 \le i \le k, 1 \le k \le n+1\} \tag{3}$$

where data are used or computed, and a dependency matrix which consists of a set of constant dependency vectors

$$D = \begin{bmatrix} \vec{e}_b^{\,3} & \vec{e}_a^{\,3} & \vec{e}_c^{\,3} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

each of them representing a data dependency corresponding to the one of the variables b,a and c, respectively.

The space of initial computations of Algorithm_1, $P_{in} = \{P_{in}(a) \cup P_{in}(b) \cup P_{in}(c)\}$ is defined as follows

$$\begin{array}{rcl} P_{in}(a) & = & \{(i,0,k) \mid 1 \le i \le k, 1 \le k \le n+1\} \\ P_{in}(b) & = & \{(0,1,k) \mid 1 \le k \le n+1\} \\ P_{in}(c) & = & \{(i,1,k) \mid 1 \le i \le n+1\} \end{array} \tag{5}$$

The bidirectional SA that implements Algorithm_1 is obtained by projecting computational structure of the algorithm (D, P_{int}) along the projection vector $\vec{\mu} = [1\ 0\ 1]^T$ (see for example [5]-[7]). Such the array is not optimal with respect to the number of processing elements (PE) for a given problem size. In order to optimize the array, we have to accommodate both the inner computation space P_{int} and the space of initial computations P_{in} to the projection direction $\vec{\mu}$. This is achieved by mapping $H = (F, G)$ (see for example [7]-[9]) which maps P_{int} into P_{int}^*, i.e.

$$P_{int} \overset{T}{\longmapsto} P_{int}^*$$

The mapping H is defined as

$$[u\ 1\ v]^T = F \cdot [i\ 1\ k]^T + G = \begin{bmatrix} 1 & 0 & 0 \\ 071 & 0 & \\ 1 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} i \\ 1 \\ k \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} = \begin{bmatrix} i \\ 1 \\ k+i-1 \end{bmatrix} \tag{6}$$

for each $i = 1,2,,k$ and $k = 1,2,,n+1$. Here $[i\ 1\ k]^T$ is an index point in the space P_{int}, while $[u\ 1\ v]^T$ is an index point in the accommodated space P_{int}^*. Similarly, the space of initial computations P_{in} is mapped into P_{in}^*. For the index points in P_{in}^* we introduce the following periodicity

$$a(i,0,k+n+1) \equiv a(i,0,k), \quad b(0,1,k+n+1) \equiv b(0,1,k), \quad c(i,1,k+n+1) \equiv c(i,1,k).$$

Now, the bidirectional SA that implements Algorithm_1 is obtained by mapping computational structure (P_{int}^*, D) using transformation matrix $S_{2\times 3}$ (see for example [5]), i.e.

$$S: \ (P_{int}^*, D) \to (\overline{P}_{int}, \Delta)$$

where $\overline{P}_{int}$ determines the (x,y) positions of the processing elements in the obtained array, while Δ defines communication links between the PEs and the direction of data flow. Matrix

S is called a valid transformation matrix and it is determined for each allowable projection direction separately. The matrix S is not uniquely defined for a given projection direction. More about the criteria for determining valid transformation matrices one can find in [12]. One of the possible transformation matrices for the direction $\vec{\mu} = [1\ 0\ 1]^T$ is

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

The (x,y) positions of the PEs in the obtained SA and data schedule at the beginning of the computation are given by the following formulas:

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

for each $i = 1,2,,k$ and $k = 1,2,,n+1$, where

$$\bar{n} = \begin{cases} n+, & \text{if } n \text{ is even} \\ n, & \text{if } n \text{ is odd} \end{cases}$$

while w is greater of the integers from the set $\{0,1\}$ that satisfies the condition

$$-2(i-1) + w\bar{n} < 0, \qquad \text{if } i = 1 \Rightarrow w = 0.$$

The communication links between the PEs and the directions of data flow are determined from

$$\Delta = S \cdot D = \begin{bmatrix} \vec{e}_b^{\,2} & \vec{e}_a^{\,2} & \vec{e}_c^{\,2} \end{bmatrix} = \begin{bmatrix} -1‘ & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}.$$

The bidirectional SA and data schedule at the beginning of the computation for n=3 are presented in Fig. 1. The structure of the PE is shown in Fig. 2. It is a cell which consists of two latches, L_B and L_C, an adder and a comparator.

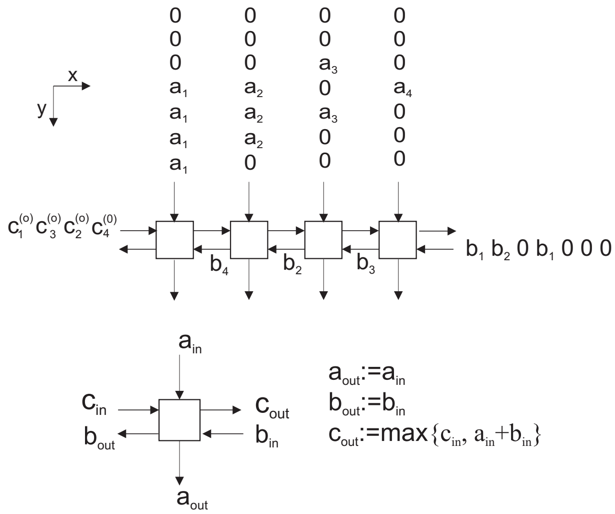

Fig. 1. The bidirectional systolic array and data schedule at the beginning of the computation, for $n = 3$

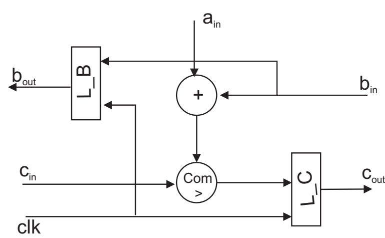

Fig. 2. The structure of the processing element

3 Performances of the basic SA

Various performance parameters can be used to measure the features of the synthesized SA. Here we will use: running (or execution time), number of processing elements, speed-up and efficiency.

The running time, T_{tot}, of the systolic algorithm includes: Tin, time required for the data elements to enter the PE where the first computation takes place; T_{exe}, time required to execute all the computations; T_{out}, time required for data elements to exit the array from the PE where the last computation finishes, i.e.

$$T_{tot} = T_{in} + T_{exe} + T_{out}.$$

In our case we have

$$T_{in} = n, \qquad T_{exe} = n+1, \qquad T_{out} = n,$$

which means that the total running time for computing one row of matrix $\bar{M} = (\bar{a}_{ri})$, $1 \le r \le m$, $2 \le i \le n+1$ is $T_{tot} = 3n+1$. Since this computation is repeated $m-1$ times the time needed to compute all rows of matrix $\bar{M}$ is equal to

$$T_t = (m-1)(3n+1).$$

The number of processing elements in the obtained SA is

$$\Omega = n+1.$$

It is an optimal number of PEs for a given problem size. This can be concluded from the fact that computing of element $\overline{a}_{r,n+1}$, for each r, requires $n+1$ calculations of the type $\max\{c, a+b\}$.

If T_1 is the execution time of Algorithm_1 on a uniprocessor system, then the speed-up of the systolic array is defined as

$$S_n = \frac{T_1}{T_t}.$$

In our case we have

$$T_1 = \frac{(m-1)(n+1)(n+2)}{2},$$

so,

$$S_n = \frac{(n+1)(n+2)}{2(3n+1} \approx O\left(\frac{n}{6}\right).$$

The efficiency of the systolic array is defined as

$$E_n = \frac{S_n}{\Omega}.$$

In our case we have

$$E_n = \frac{n+2}{2(3n+1)} \approx O\left(\frac{1}{6}\right).$$

4 Conclusion

The solution of optimal investment problem has been discussed in this paper. To speed-up the computation a spcial purpose parallel architecture, namely linear systolic array, was synthesized. The obtained array is optimal with respect to a problem size.

References

[1] S. I. Zuhovickij, L. I. Advejeva, *Linear and convex programming*, Fiz-Mat, Moscow, 1957.

[2] M. K. Stojčev, *CISC, RISC and DSP processors*, Fculty of Electronic Engineering, Niš, 1997.

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

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

[5] S. G. Sedukhin, *The designing and analysis of systolic algorithms and structures*, Programming, 2 (1990), 20-40.

[6] D. I. MOLDOVAN, *Parallel processing: From applications to systems*, Morgan Kaufman Publishers, San Mateo, 1993.

[7] 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. Electrotehn. Math., 4 (1999), 35-40.

[8] E. I. MILOVANOVIĆ, G. V. MILOVANOVIĆ, I. Ž. MILOVANOVIĆ, D. MILOSAVLJEVIĆ, *Designing hexagonal systolic arrays by composite mappings*, Facta Universitatis, Ser. Math. Inform., 12 (1997), 283-296.

[9] T. I. TOKIĆ, I. Ž. MILOVANOVIĆ, D. M. RANDJELOVIĆ, E. I. MILOVANOVIĆ, *Determining VLSI array size for one class of nested loop algorithms*, *Advances in Computer and Information Sciences*, Antalia'98, (U. Gudukbay, T. Dagar, A. Gursay, E. Gelembe, eds.), IDS Press, 1998, 389-396.

[10] E. I. MILOVANOVIĆ, M. K. STOJČEV, N. M. NOVAKOVIĆ, I. Ž. MILOVANOVIĆ, T. I. TOKIĆ, *Matrix-vector multiplication on fixed-size linear systolic array*, Comput. Math. Appl., 40 (2000) 1189-1203.

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

[12] M. P. BEKAKOS, E. I. MILOVANOVIĆ, N. M. STOJANOVIĆ, T. I. TOKIĆ, I. Ž. MILOVANOVIĆ, I. Z. MILENTIJEVIĆ, *Transformation matrices for systolic array synthesis*, J. Elecroteh. Math., Vol. 7, 1 (2002), 9-15.