Faculty of Sciences and Mathematics, University of Niš, Serbia
Available at: http://www.pmf.ni.ac.yu/filomat

Filomat **23:1** (2009), 1–12

FINDING MINIMUM COST SPANNING TREE ON BIDIRECTIONAL LINEAR SYSTOLIC ARRAY*

E. I. Milovanović, M. P. Bekakos, I. Ž. Milovanović and B. M. Randjelović

Abstract

In this paper a bidirectional linear systolic array (BLSA) that computes minimum cost spanning tree (MCST) of a given graph is designed. We present a synthesis procedure, based on data dependencies and space-time transformations of index space, to design BLSA with optimal number of processing elements (PEs) for a given problem size. The execution time is minimized for that number of PEs. Explicit mathematical formulas for systolic array synthesis are derived. We compare performances of the BLSA with the unidirectional linear systolic array (ULSA).

1 Introduction

A graph often contains redundancy in that there can be multiple paths between two vertices. This redundancy may be desirable, for example to offer alternative routes in the case of breakdown or overloading of an edge (road, connection, phone-line) in a network. However, we often require the cheapest sub-network, that connects the vertices of a given graph. The minimum cost spanning tree (MCST) of a graph defines the cheapest subset of edges that keeps the graph in one connected component. MCST problem is perhaps the simplest and certainly one of the most central models in the field of network optimization. The problem is important both in its practical and theoretical applications. It arises in a number of applications, both as a stand-alone problem and as a sub-problem in more complex problem settings. For example, telephone companies or cable-TV companies are particulary interested in MCST, because the minimum spanning tree of a set of sites defines the

*Work supported by the Serbian Council of Science and Environmental Protection, grant 144034.

2000 *Mathematics Subject Classifications.* 68R10.

Key words and Phrases. Spanning tree, Systolic array, Systolic algorithm.

Received: August 10, 2007

Communicated by Dragan S. Djordjević

wiring scheme that connects the sites using as little wire as possible. It is mother of all network design problems. A less obvious application is that the minimum spanning tree can be used to approximately solve the travelling salesman's problem. A convenient formal way of defining this problem is to find the shortest path that visits each point at least once.

The problem of finding MCST can be formulated as follows. Let $\Gamma = (V, E)$ be a weighted directed or undirected graph, where $V = \{1, 2, ..., n\}$ is a set of n vertices, E is a set of edges, and $D = (d_{ij})$ is an $n \times n$ weight matrix, i.e. d_{ij} is the cost of edge $(i, j) \in E$ determined as

$$d_{ij} = \begin{cases} d_{ij}, & \text{if } (i,j) \in E, \\ +\infty, & \text{if } (i,j) \notin E, \\ 0, & \text{if } i = j, \end{cases}$$

for each $i = 1, 2, ..., n$ and $j = 1, 2, ..., n$. The problem is to find a rooted spanning tree $\Gamma' = (V, E')$ where E' is a subset of E such that the sum of d_{ij} for all (i, j) in E' is minimized. The spanning tree is defined as a graph which connects, without any cycle, all vertices with $n-1$ arcs, i.e. each vertex, except the root, has one and only one incoming arc. Note that a minimum cost spanning tree is not necessarily unique. There is a variety of algorithms to solve MCST problem efficiently. Czech scientist Otakar Boruvka formulated the first efficient solution of MCST problem as early as 1926 (see [1, 2]). Now, there are two algorithms commonly used, Prim's algorithm [3] and Kruskal's algorithm [4]. In this paper we use Maggs-Plotkin minimal spanning tree algorithm [5] which is closely related to Warshall transitive closure [8] and Floyd shortest path algorithm [9]. It is assumed that edge costs are unique (distinct) [5], [7].

Maggs-Plotkin algorithm for finding MCST starts from the matrix $D^{(0)} = D$ and computes a series of matrices $D^{(k)} = (d_{ij}^{(k)})$, $k = 1, 2, ..., n$, according to the following:

Algorithm_1
for $k := 1$ **to** n **do**
for $i := 1$ **to** n **do**
for $j := 1$ **to** n **do**
$d_{ij}^{(k)} := \min\{d_{ij}^{(k-1)}, \max\{d_{ik}^{(k-1)}, d_{kj}^{(k-1)}\}\}$

Once $D^{(n)}$ is obtained, it is easy to determine the tree edges, namely an edge (i, j), $(i, j) \in E$, is added to MCST if and only if $d_{ij}^{(n)} = d_{ij}^{(0)}$ for $i = 1, 2, ..., n$ and $j = 1, 2, ..., n$. If weights d_{ij} are in the range $[0, 1]$ and min and max operators are switched, than Maggs-Plotkin algorithm can be used to compute max-min transitive closure of fuzzy similarity matrix (see, for example [6]).

Maggs and Plotkin [5] consider implementation of the proposed algorithm on an $n \times n$ mesh-connected computer in $O(n)$ steps. In [7] this algorithm was used to compute MCST on a linear array with reconfigurable pipelined bus system. The problem is solved in $O(\log n)$ time by using n^3 processors. In this paper we design

a bidirectional linear systolic array (BLSA) with two-dimensional I/O links with n processing elements (PEs) that computes $D^{(n)}$ for $O(2n^2)$ steps. Here we are only interested in finding matrix $D^{(n)}$. The comparison and computation of the cost can be performed by the host computer. Our array is pure systolic in the sense that it has only local interconnections and very simple processing elements. The methodology used in this paper represents the modification of the procedure based on the 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. This graph is then projected along defined directions to obtain the target architecture. Our modification enables obtaining systolic array with optimal number of PEs for a given problem size. To overcome the problem of irregularity of the above minimal spanning tree algorithm, we partition the problem into regular two-dimensional instances which can be computed on BLSA. The final result is obtained by repeating the computation appropriate number of times. The designed BLSA has optimal number of processing elements (PE) for a given problem size and minimal execution time for that number of PEs. Obtained results will be compared with the one obtained for the unidirectional linear systolic array (ULSA), proposed in [10].

2 The systolic algorithm

In order to obtain an algorithm which is suitable for the synthesis of the BLSA we partition the computations in Algorithm_1 into two-dimensional entities $D^{(k)} = (d_{ij}^{(k)})$ for some fixed $k = 1, 2, \ldots, n$. The computation of $D^{(k)}$ depends only on $D^{(k-1)}$. This means that the computations of $D^{(1)}, \ldots, D^{(n)}$ can be performed successively, which is very important for our approach. It is enough to synthesize the BLSA that computes $D^{(k)}$ for some fixed k, and then use it to compute $D^{(n)}$ by repeating the computations n times. Without lost of generality, we take $k = 1$. To ease the presentation we use the following notation

$$a(i,0,1) = d_{i1}^{(0)}, b(0,j,1) = d_{1j}^{(0)}, c(i,j,0) = d_{ij}^{(0)}, c(i,j,1) = d_{ij}^{(1)}, \tag{1}$$

for $i = 1, 2, ..., n$ and $j = 1, 2, ..., n$.

The corresponding systolic algorithm has the following form

Algorithm_2
for $i := 1$ **to** n **do**
for $j := 1$ **to** n **do**
$a(i,j,1) := a(i,j-1,1)$
$b(i,j,1) := b(i-1,j,1)$
$c(i,j,1) := \mathbf{min}\{c(i,j,0), \mathbf{max}\{a(i,j,1), b(i,j,1)\}\}$

The index or iteration space of Algorithm_2 is given by

$$P_{int} = \{(i,j,1) | 1 \leq i \leq n, 1 \leq j \leq n\}. \tag{2}$$

The data dependency vectors $\vec{e}_b^3$, $\vec{e}_a^3$, $\vec{e}_c^3$ of variables $b(0,j,1)$, $a(i,0,1)$ and $c(i,j,0)$ in Algorithm_2 are determined as $\vec{e}_b^3 = [i\ j\ 1]^T - [i-1\ j\ 1]^T = [1\ 0\ 0]^T$, $\vec{e}_a^3 = [i\ j\ 1]^T - [i\ j-1\ 1]^T = [0\ 1\ 0]^T$ and $\vec{e}_c^3 = [i\ j\ 1]^T - [i\ j\ 0]^T = [0\ 0\ 1]^T$, respectively. Thus, the dependency matrix of Algorithm_2 is defined by

$$Q = \left[\ \vec{e}_b^{\,3} \quad \vec{e}_a^{\,3} \quad \vec{e}_c^{\,3}\ \right] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3}$$

The systolic algorithm Algorithm_2 is completely defined by a pair (P_{int}, Q), i.e. by directed graph $G = (P_{int}, Q)$, where P_{int} represents a set of vertices, while matrix Q, i.e. its column vectors, represent directions of edges between adjacent nodes. This graph is placed in a three-dimensional orthogonal coordinate system, generated by the vectors $\vec{i} = [1\ 0\ 0]^T$, $\vec{j} = [0\ 1\ 0]^T$ and $\vec{k} = [0\ 0\ 1]^T$. This is also a graph of two-dimensional systolic array, placed in the plane $k = 1$, suitable for the implementation of Algorithm_2 (however, not efficiently enough). The possible projection direction vectors that can be used to obtain 1D SAs with near-neighbor communications are $\vec{\mu} = [\mu_1\ \mu_2\ \mu_3]^T$, where $\mu_i \in \{-1, 0, 1\}$, $i = 1, 2, 3$ (see [11]). There are $3^3 = 27$ projection direction vectors in total. The direction $\vec{\mu} = [0\ 0\ 0]^T$ is excluded for obvious reasons. Further, the directions $\vec{\mu}$ and $-\vec{\mu}$ give the same projections. Thus, there are only 13 different projection direction vectors to be considered. It is obvious that $\mu_3 = 0$, since graph $G = (P_{int}, Q)$ is placed in the plane $k = 1$. Therefore, projection direction vectors that map Algorithm_2 into 1D SA are $\mu = [1\ 0\ 0]^T$, $\mu = [0\ 1\ 0]^T$, $\mu = [1\ -1\ 0]^T$ and $\mu = [1\ 1\ 0]^T$. The directions $\mu = [1\ 0\ 0]^T$, and $\mu = [0\ 1\ 0]^T$ give orthogonal projections and the corresponding SAs are static, meaning that one of the variables is resident in the array. By the direction $\mu = [1\ -1\ 0]^T$ the ULSA, considered in [10], is obtained. Thus, only the direction $\mu = [1\ 1\ 0]^T$ can be used to obtain BLSA.

By choosing suitable linear transformation T, from the set of valid transformations that correspond to $\mu = [1\ 1\ 0]^T$, the graph $G = (P_{int}, Q)$ is mapped onto BLSA so that the dependencies between the computations are supported by the physical connections of the array. From a set of valid transformations we choose arbitrarily (see, for example [12])

$$T = \begin{bmatrix} \vec{\Pi} \\ -- \\ S \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ - & - & - \\ -1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{4}$$

The (x, y) coordinates of the PEs in the BLSA that implements Algorithm_2 are obtained by mapping index-space P_{int}, defined in (2), using matrix S from (4). The communication links between the PEs are obtained by mapping dependency matrix Q, defined in (3), using the same matrix S. However, the obtained BLSA is not optimal with respect to the number of PEs for a given problem size. According to Theorem defined in [13], it can be concluded that number of PEs in this BLSA is $\Omega = 2n - 1$. In order to obtain SA with optimal number of PEs, i.e. $\Omega = n$, for a

given problem size, we have to accommodate, i.e. to reorder, the index space P_{int} of Algorithm_2, to the direction $\mu = [1\ 1\ 0]^T$ such that exactly n index points of the reordered index space are placed on the lines parallel to the direction $\mu = [1\ 1\ 0]^T$.

The accommodation is possible due to the fact that row elements of matrix C can be computed in any order. Rows can also be computed in any order. This means the computations in Algorithm_2 can be performed at arbitrary permutation $p_1, p_2, \ldots, p_n$ of index variable i or j, where $\{1, 2, ..., n\}$ is the basic permutation (see [14]). This mapping strongly depends on $\vec{\mu}$. It is defined as

$$F = (H, \vec{q}), \quad F:\ P_{int} \longmapsto P^*_{int}, \quad P^*_{int} = \{(u, v, 1)\}$$

where H is a scalar matrix of order 3×3 while $\vec{q} = [q_1\, q_2\, q_3]^T$ is a vector that ensures that accommodated space P^*_{int} is placed in the first octant. The space P^*_{int} is obtained according to

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = H \cdot \vec{p} + \vec{q}, \quad \vec{p} \in P_{int}$$

Since the direction $\mu = [1\ 1\ 0]^T$, has both $\mu_1 = 1 \neq 0$ and $\mu_2 = 1 \neq 0$, the accommodation of P_{int} is possible on both index variable i and j. When the accommodation is performed on index variable i, mapping $F = (H, \vec{q})$ is of the form (see [14])

$$H = \begin{bmatrix} \vec{\mu} & \vec{j} & \vec{k} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad \vec{q} = \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix}, \tag{5}$$

while when it is performed on index variable j, it is of the form

$$H = \begin{bmatrix} \vec{j} & \vec{\mu} & \vec{k} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad \vec{q} = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}. \tag{6}$$

When F is defined by (5), the accommodated space, P^*_{int} is obtained according to

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = H \cdot \vec{p} + \vec{q} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} i \\ j \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix} = \begin{bmatrix} i \\ i + j - 1 \\ 1 \end{bmatrix}, \tag{7}$$

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

Similarly, when F is defined by (6) P^*_{int} is obtained from

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = H \cdot \vec{p} + \vec{q} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} i \\ j \\ 1 \end{bmatrix} + \begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} i + j - 1 \\ j \\ 1 \end{bmatrix}. \tag{8}$$

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

Without luck of generality, we will prove that P^*_{int} obtained according to (7) is really accommodated to the direction $\vec{\mu} = [1\ 1\ 0]^T$, i.e. that exactly n points of P^*_{int} lie on the line parallel to $\vec{\mu} = [1\ 1\ 0]^T$. Let $\vec{p}^*_1 = [i+1\ i+j\ 1]^T$ and $\vec{p}^*_2 = [i\ i+j-1\ 1]^T$ be the position vectors of points from P^*_{int}, where i is fixed at some constant value and $j = 1, 2, \ldots, n-1$. Since

$$\vec{p}^*_2 - \vec{p}^*_1 = \vec{\mu},$$

we conclude that these n points are positioned on the line parallel to $\vec{\mu}$. All these points are mapped into a single point in the processor plane. Since $i = 1, 2, \ldots, n$, there are n such lines in total, which implies that projection of P^*_{int} has exactly n points, i.e. there will be n PEs in the BLSA.

The new index-spaces $P^*_{int} = \{(u, v, 1)\}$, determined from (7) and (8),

$$P^*_{int} = \{(i, i+j-1, 1) | 1 \le i \le n\,, 1 \le j \le n\} \tag{9}$$

and

$$P^*_{int} = \{(i+j-1, j, 1) | 1 \le i \le n\,, 1 \le j \le n\} \tag{10}$$

define two new regular directed graphs, $G^* = (P^*_{int}, Q)$, where Q is given by (3). The systolic algorithms corresponding to these graphs are equivalent to Algorithm_2 and have the following form

Algorithm_3
for $i := 1$ **to** n **do**
for $j := 1$ **to** n **do**
$a(i, i+j-1, 1) := a(i, i+j-2, 1)$
$b(i, i+j-1, 1) := b(i-1, i+j-1, 1)$
$c(i, i+j-1, 1) :=$ **min**$\{c(i, i+j-1, 0),$
max$\{a(i, i+j-1, 1), b(i, i+j-1, 1)\}\}$

where $b(0, j+n, 1) = b(0, j, 1)$, $c(i, j+n, 0) = c(i, j, 0)$, $c(i, j+n, 1) = c(i, j, 1)$, for $i = 1, 2, ..., n$ and $j = 1, 2, ..., n$.

Algorithm_4
for $i := 1$ **to** n **do**
for $j := 1$ **to** n **do**
$a(i+j-1, j, 1) := a(i+j-1, j-1, 1)$
$b(i+j-1, j, 1) := b(i+j-2, j, 1)$
$c(i+j-1, j, 1) :=$ **min**$\{c(i+j-1, j, 0),$
max$\{a(i+j-1, j, 1), b(i+j-1, j, 1)\}\}$

where $a(i+n, 0, 1) = a(i, 0, 1)$, $c(i+n, j, 0) = c(i, j, 0)$, $c(i+n, j, 1) = c(i, j, 1)$, for $i = 1, 2, ..., n$ and $j = 1, 2, ..., n$.

Since the upper bounds of index variables i and j are equivalent ($1 \le i \le n$ and $1 \le j \le n$), the BLSAs synthesized according to Algorithm_3 and Algorithm_4 have the same space and time features. Therefore in the text that follows we will consider only Algorithm_3.

3 The BLSA synthesis

The BLSA that implements Algorithm_3 is obtained by mapping directed graph $G^* = (P^*_{int}, Q)$, using space transformation matrix S defined by (4). The (x, y)-coordinates of PEs in the BLSA are obtained according to

$$PE \mapsto \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} j-1 \\ 1 \end{bmatrix}, \quad 1 \le j \le n. \tag{11}$$

The communication links between PEs in the BLSA are implemented along the column vectors of the matrix

$$\Delta = S \cdot Q = \left[\vec{e}_b{}^2 \ \vec{e}_a{}^2 \ \vec{e}_c{}^2\right] = \begin{bmatrix} -1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{12}$$

The space of initial computations of Algorithm_3, $P_{in} = P_{in}(a) \cup P_{in}(b) \cup P_{in}(c)$, is defined by index sets

$$\begin{aligned} P_{in}(a) &= \{(i, 0, 1) \,|\, 1 \le i \le n\} \\ P_{in}(b) &= \{(0, i+j-1, 1) \,|\, 1 \le i \le n,\ 1 \le j \le n\} \\ P_{in}(c) &= \{(i, i+j-1, 0) \,|\, 1 \le i \le n,\ 1 \le j \le n\}. \end{aligned}$$

In order to obtain correct timing of the input data items, the space P_{in} have to be reordered. The reordering is performed by the timing function $t(\vec{p}) = \vec{\pi} \cdot \vec{p} + \alpha$, for each $\vec{p}$, where $\vec{p}$ is a position vector of the point in P_{in}, $\vec{\pi} = [1\ 1\ 1]$ is row-vector of matrix T, defined in (4), and α is a constant, that provides that the first computation is executed at the point $(1, 1, 1)$ of the space P^*_{int} (see [14]). In our case, $t(\vec{p}) = u + v + w - 3$, for each $\vec{p} = [u\ v\ w]^T$. Reordering of P_{in} is performed according to the following

$$\vec{p}_\nu{}^* = \vec{p}_\nu - (t(\vec{p}_\nu) + 1)\vec{e}_\nu{}^3, \qquad \nu \in \{a, b, c\}. \tag{13}$$

The reordered space of initial computations $P^*_{in} = P^*_{in}(a) \cup P^*_{in}(b) \cup P^*_{in}(c)$, is defined by the sets

$$\begin{aligned} P^*_{in}(a) &= \{(i, 1-i, 1) \,|\, 1 \le i \le n\} \\ P^*_{in}(b) &= \{(2-i-j, i+j-1, 1) \,|\, 1 \le i \le n,\ 1 \le j \le n\} \\ P^*_{in}(c) &= \{(i, i+j-1, 3-2i-j) \,|\, 1 \le i \le n,\ 1 \le j \le n\}. \end{aligned} \tag{14}$$

The positions of input data items at the beginning of the computation on the BLSA are obtained by mapping P^*_{int}, defined by (14) using matrix S from (4). However, as a pipelining-period during implementation of Algorithm_3 on the obtained BLSA is $\lambda = |\vec{\pi} \cdot \vec{\mu}| = 2$, each PE is active in every other clock cycle. Therefore the active execution time is not as minimal as possible. In order to minimize the execution time we will take over the method from [15], defined for the synthesis of two-dimensional hexagonal SA for matrix product. We will slightly modify this method to accommodate it to our problem.

First, we introduce some notations.

1) Parameter $\bar{n}$ is an integer defined as

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

2) Parameters r_1 and r_2 are determined for all pairs (i, j) as greater of the integers from the set $\{0, 1\}$, such that the following is satisfied

$$\begin{aligned} &-2(i-1) + r_1\bar{n} < 0, \quad i = 1 \quad \Rightarrow \quad r_1 = 0, \\ &-2(i-1) - (j-1) + (r_1 + r_2)\bar{n} \leq 0. \end{aligned}$$

Parameter r_1 is determined first.

Now we can determine an optimal data schedule at the beginning of the computation in the BLSA that yields minimal execution time. The (x, y) positions of input data items at the beginning of the computation are obtained from the following formulas

$$\begin{aligned} a(i,0,1) &\mapsto \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1-2i \\ 1 \end{bmatrix} + (r_1+r_2)\bar{n} \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \\ b(0,i+j-1,1) &\mapsto \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 2i+2j-3 \\ 1 \end{bmatrix} + (r_1+r_2)\bar{n} \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \\ c(i,i+j-1,0) &\mapsto \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} j-1 \\ 3-2i-j \end{bmatrix} + (r_1+r_2)\bar{n} \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \end{aligned} \tag{15}$$

for each $i = 1, 2, ..., n$ and $j = 1, 2, ..., n$.

Data schedule at the beginning of the computation on the BLSA, for the case n = 3, is depicted in Fig.1. The structure of the PE is depicted in Fig.2. The PE consists of two comparators (Com) and three latches labelled as L_A, L_B and L_C.

4 Performance analysis

According to (11) is not difficult to conclude that obtained BLSA has $\Omega = n$ PEs. This is optimal number of PEs, for a given problem size.

Assume that time needed to perform an operation of type **max** and **min** represents one time unit. Denote with t_{in} initialization time, t_{exe} execution time, t_{out} output time, and t_{tot} the total execution time of Algorithm_3 on the BLSA. According to (15) we have that $t_{in} = n - 1$, $t_{exe} = \bar{n}$ and $t_{out} = n - 1$. Since $t_{tot} = t_{in} + t_{exe} + t_{out}$, we have $t_{tot} = 2n + \bar{n} - 2$. Recall that we have designed BLSA that implements Algorithm_3, i.e. computes $D^{(1)} = (d_{ij}^{(1)})$. Matrix $D^{(n)}$ is obtained by computing $D^{(1)}, D^{(2)}, \ldots, D^{(n-1)}, D^{(n)}$, such that elements of $D^{(k)} = (d_{ij}^{(k)})$ are used as inputs for computing $D^{(k+1)} = (d_{ij}^{(k+1)})$, for $k = 1, 2, \ldots, n-1$. During the computation the output time of k-th iteration is overlapped with the input time of the $(k+1)$-st iteration. This is illustrated in Table 1 for the case $n = 3$. In Table 1, the first two clock cycles represent the initialization time of the first iteration. During the cycles 6 and 7 (11 and 12) the output and initialization time of two

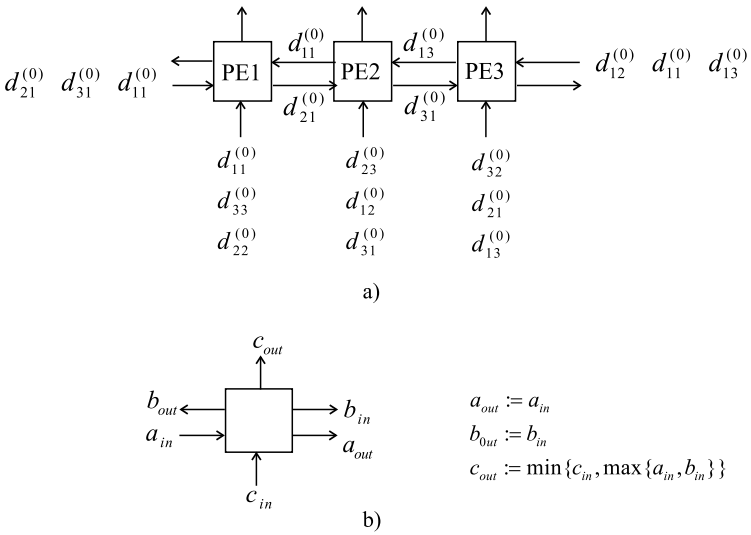

Figure 1: a) Data flow in the BLSA for the case $n = 3$. b) Functional property of the PE.

successive iterations are overlapped. Having this in mind, the total time required to compute matrix $D^{(n)}$, on the BLSA is $T_{tot} = n(n + \bar{n} + 1)$.

The efficiency of the BLSA is

$$E = \frac{n}{n + \bar{n} - 1} \approx \frac{1}{2} .$$

In Table 2 number of PEs, total execution time and the efficiency of the designed BLSA and ULSA proposed in [10] for computing MCST are given. It can be concluded that both arrays have the same number of PEs, but the execution time of BLSA is less than that of ULSA. For small n efficiency of both arrays is the same, i.e. $\frac{1}{2}$. However, for greater n the efficiency of the ULSA tends to 0.33.

5 Conclusion

A problem of computing MCST of a given graph on the bidirectional linear systolic array was considered. Maggs-Plotkin algorithm [5] was used to compute matrix $D^{(n)}$ from a cost matrix D. Once $D^{(n)}$ is obtained, it is easy to determine the tree edges, namely an edge (i, j), $(i, j) \in E$, is added to MCST if and only if $d_{ij}^{(n)} = d_{ij}$ for $i = 1, 2, ..., n$ and $j = 1, 2, ..., n$. A problem is partitioned into two-dimensional entities. Then the BLSA that computes one entity is designed. The final result is obtained iteratively. By the appropriate definition of data schedule an overlapping of the output and input time of two successive iterations is achieved. The BLSA

Clk	PE1	PE2	PE3
1	$0 = \min(0, \max(d_{31}^{(0)}, 0))$		$0 = \min(0, \max(0, d_{11}^{(0)}))$
2	$0 = \min(0, \max(d_{21}^{(0)}, 0))$	$0 = \min(0, \max(d_{31}^{(0)}, d_{11}^{(0)}))$	$0 = \min(0, \max(0, d_{13}^{(0)}))$
3	$d_{11}^{(1)} = \min(d_{11}^{(0)}, \max(d_{11}^{(0)}, d_{11}^{(0)}))$	$d_{23}^{(1)} = \min(d_{23}^{(0)}, \max(d_{21}^{(0)}, d_{13}^{(0)}))$	$d_{32}^{(1)} = \min(d_{32}^{(0)}, \max(d_{31}^{(0)}, d_{12}^{(0)}))$
4	$d_{33}^{(1)} = \min(d_{33}^{(0)}, \max(d_{31}^{(0)}, d_{13}^{(0)}))$	$d_{12}^{(1)} = \min(d_{12}^{(0)}, \max(d_{11}^{(0)}, d_{12}^{(0)}))$	$d_{21}^{(1)} = \min(d_{21}^{(0)}, \max(d_{21}^{(0)}, d_{11}^{(0)}))$
5	$d_{22}^{(1)} = \min(d_{22}^{(0)}, \max(d_{21}^{(0)}, d_{12}^{(0)}))$	$d_{31}^{(1)} = \min(d_{31}^{(0)}, \max(d_{31}^{(0)}, d_{11}^{(0)}))$	$d_{13}^{(1)} = \min(d_{13}^{(0)}, \max(d_{11}^{(0)}, d_{13}^{(0)}))$
6	$0 = \min(0, \max(d_{32}^{(1)}, d_{11}^{(0)}))$	$0 = \min(0, \max(d_{21}^{(0)}, d_{13}^{(0)}))$	$0 = \min(0, \max(d_{31}^{(0)}, d_{21}^{(1)}))$
7	$0 = \min(0, \max(d_{22}^{(1)}, d_{13}^{(0)}))$	$0 = \min(0, \max(d_{32}^{(1)}, d_{21}^{(1)}))$	$0 = \min(0, \max(d_{21}^{(0)}, d_{23}^{(1)}))$
8	$d_{11}^{(2)} = \min(d_{11}^{(1)}, \max(d_{12}^{(1)}, d_{21}^{(1)}))$	$d_{23}^{(2)} = \min(d_{23}^{(1)}, \max(d_{22}^{(1)}, d_{23}^{(1)}))$	$d_{32}^{(2)} = \min(d_{32}^{(1)}, \max(d_{32}^{(1)}, d_{22}^{(1)}))$
9	$d_{33}^{(2)} = \min(d_{33}^{(1)}, \max(d_{32}^{(1)}, d_{23}^{(1)}))$	$d_{12}^{(2)} = \min(d_{12}^{(1)}, \max(d_{12}^{(1)}, d_{22}^{(1)}))$	$d_{21}^{(2)} = \min(d_{21}^{(1)}, \max(d_{22}^{(1)}, d_{21}^{(1)}))$
10	$d_{22}^{(2)} = \min(d_{22}^{(1)}, \max(d_{22}^{(1)}, d_{22}^{(1)}))$	$d_{31}^{(2)} = \min(d_{31}^{(1)}, \max(d_{32}^{(1)}, d_{21}^{(1)}))$	$d_{13}^{(2)} = \min(d_{13}^{(1)}, \max(d_{12}^{(1)}, d_{23}^{(1)}))$
11	$0 = \min(0, \max(d_{33}^{(2)}, d_{21}^{(1)}))$	$0 = \min(0, \max(d_{22}^{(1)}, d_{23}^{(1)}))$	$0 = \min(0, \max(d_{32}^{(1)}, d_{31}^{(2)}))$
12	$0 = \min(0, \max(d_{23}^{(2)}, d_{23}^{(1)}))$	$0 = \min(0, \max(d_{33}^{(2)}, d_{31}^{(2)}))$	$0 = \min(0, \max(d_{22}^{(1)}, d_{33}^{(2)}))$
13	$d_{11}^{(3)} = \min(d_{11}^{(2)}, \max(d_{13}^{(2)}, d_{31}^{(2)}))$	$d_{23}^{(3)} = \min(d_{23}^{(2)}, \max(d_{23}^{(2)}, d_{33}^{(2)}))$	$d_{32}^{(3)} = \min(d_{32}^{(2)}, \max(d_{33}^{(2)}, d_{32}^{(2)}))$
14	$d_{33}^{(3)} = \min(d_{33}^{(2)}, \max(d_{33}^{(2)}, d_{33}^{(2)}))$	$d_{12}^{(3)} = \min(d_{12}^{(2)}, \max(d_{13}^{(2)}, d_{32}^{(2)}))$	$d_{21}^{(3)} = \min(d_{21}^{(2)}, \max(d_{23}^{(2)}, d_{31}^{(2)}))$
15	$d_{22}^{(3)} = \min(d_{22}^{(2)}, \max(d_{23}^{(2)}, d_{32}^{(2)}))$	$d_{31}^{(3)} = \min(d_{31}^{(2)}, \max(d_{33}^{(2)}, d_{31}^{(2)}))$	$d_{13}^{(3)} = \min(d_{13}^{(2)}, \max(d_{13}^{(2)}, d_{33}^{(2)}))$

Table 1: Computations performed in the PEs while computing MCST of the graph for $n = 3$.

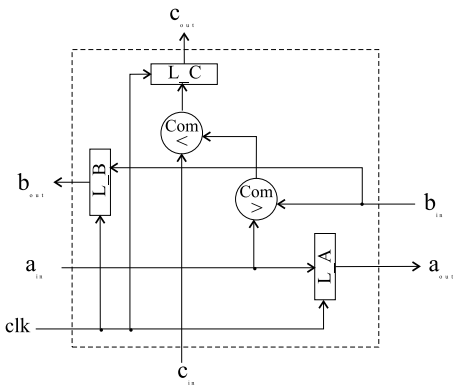

Figure 2: The structure of the PE.

SA	Ω	T_{tot}	E
BLSA	n	$n(n+\bar{n}+1)$	$\frac{n}{n+\bar{n}-1}$
ULSA	n	$n(3n-2)$	$\frac{n}{3n-2}$

Table 2: Performances of the BLSA and ULSA

consists of n PEs, which is optimal number for a given problem size. The problem is solved in $2n^2+n$ steps. The efficiency of the array is 0.5.

References

[1] O. Boruvka, About a certain minimal problem, *Prace Mor. Prirodoved. Spol. v Brne III*, **3**,(1926), 37–58.

[2] J. Nešetril , A few remarks on the history of MST-problem, *Archivum Mathematicum (Brno)*, **33**,(1997), 15–22.

[3] R. C. Prim , The shortest connecting network and some generalizations, *Bell Syst. Tech. Journal*, **36**,(1957), 1389–1401.

[4] J. B. Kruskal , On the shortest spanning subtree of a graph and the travelling salesman problem, *Proc. Amer. Math. Soc*, **7**,(1956), 48–50.

[5] B. M. Maggs, S. A. Plotkin, Minimum-cost spanning tree as a path-finding problem, *Inform. Process. Lett.*, **6**, (1988), 291-293.

[6] S. Kundu, An optimal $O(N^2)$ algorithm for computing the min-transitive closure of a weighted graph, *Inform. Process. Lett.*, **74**, (2000), 215-220.

[7] K. Li, Y. Pan, M. Hamdi, Solving graph theory problems using reconfigurable pipelined optical buses, *Parallel Comput.*, 26, (2000), 723-735.

[8] S. Warshall, A theorem on Boolean matrices, *J. ACM*, **9**, (1962), 11–12.

[9] R. W. Floyd, Algorithm 97, Shortest path, *C. ACM*, **5**,(1962), 5.

[10] E. I. Milovanović, I. Ž. Milovanović , B. M. Randjelović , Computing minimum Cost spanning tree on Linear Unidirectional Systolic Array, *Conference Proceedings: ICEST 2005 , Niš, Serbia and Montenegro*, **1**,(2005), 437-440.

[11] S. G. Sedukhin , The Designing and Analysis od Systolic Algorithms and Structures, *Programming*, **2**, (1990), 20–40.

[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.Electrotechn. Math.*, **7**, (2002), 9-15.

[13] C. N. Zhang, J. H. Weston, I.-F. Yan, Determining objective funkctions in systolic array designs, *IEEE Trans. VLSI Systems*, **2(3)**, (1994), 357-360.

[14] T. I. Tokić, I. Ž. Milovanović, D. M. Randjelović, E. I. Milovanović, VLSI array size for one class of nested loop algorithms, *Conference Proceedings: Advances in Computer and Information Sciences, Antalia, Turkey* (U. Güdükbay, T. Dagar, A. Gürsay, E. Gelembe, eds.), IDS-Press,(1998), 389-396.

[15] 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.), WIT-Press, Southampton, Boston*, (2001), 175-209.

Address:

E. I. Milovanović, I. Ž. Milovanović, B. M. Randjelović: Faculty of Electronic Engineering, A. Medvedeva 14, 18000 Niš, Serbia
E-mail: igor@elfak.ni.ac.yu

M. P. Bekakos: Department of Electrical and Computer Engineering, School of Engineering, Democritus University of Thrace, Greece