Computing Minimum Cost Spanning Tree on Linear Unidirectional Systolic Array

E. I. Milovanović[1], I. Ž. Milovanović[2], B. M. Randjelović[3]

Abstract –**A problem of finding MCST of a given graph is considered. The problem is partitioned into two parts. First, we compute a wighted matrix, $D^{(n)}$, according to weighted matrix, $D^{(0)}$, of a given graph. Second, MCST is determined by comparing matrices $D^{(n)}$ and $D^{(0)}$. To solve the first problem we designed unidirectional linear systolic array with optimal number of PEs and minimize the execution time. The secdond is performed by the host computer.**

Keywords – **Systolic array, Minimum cost spanning tree.**

I. INTRODUCTION

The minimum cost spanning tree (MCST) of a graph defines the cheapest subset of edges that keeps the graph in one connected component. The MCST problem arises in a number of applications, both as a stand-alone problem and as a sub problem in more complex problem settings. It is perhaps the simplest, and certainly one of the most central, models in the field of network optimisation.

The problem of finding MCST can be formulated as follows. Consider a connected undirected or directed graph, $G=(V,E)$, where $V=\{1,2,...,n\}$ is the set of vertices and E is the set of edges. Associated with each edge (i,j) in E is a cost d_{ij}. Thus, weighted matrix $D^{(0)}=(d_{ij}^{(0)})$ of order nxn which corresponds to graf G can be described as

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

for each $i=1,2,\ldots,n$ and $j=1,2,\ldots,n$. The problem is to find a rooted spanning tree, $G=(V,E')$ 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 nodes with n-1 arcs, i.e., each node, except the root, has one and only one incoming arc. Note that a minimum cost spanning tree is not necessarily unique. This problem can be solved by many different algorithms. The first algorithm for finding MST was developed by Czech scientist Otakar Boruvka in 1926 [1-2]. Now, there are two algorithms commonly used, Prim's algorithm and Kruskal's algorithm [3], [4]. In this paper we will use an algorithm equivalent to Warshall's [5] and Floyd's algorithms [6] for computing transitive closure and shortest path in a given graph.

[1]E. I. Milovanović, [2]I. Ž. Milovanović, [3]B. M. Randjelović are with the Faculty of Electronic Engineering, Aleksandra Medvedeva 14, 18000 Nis, Serbia and Montenegro E-mail: ema@elfak.ni.ac.yu, igor@elfak.ni.ac.yu, bane@elfak.ni.ac.yu

In order to find MCST we start from the matrix $D^{(0)}=(d_{ij}^{(0)})$ and compute a series of matrices $D^{(k)}=(d_{ij}^{(k)}), k=1,2,\ldots,n$, according to the following algorithm:

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

In the second step matrix $D^{(0)}$ is compared with matrix $D^{(n)}$. An edge (i,j), $(i,j)\in E$, is added to MCST if and only if $d_{ij}^{(n)}=d_{ij}^{(0)}$ for $i=1,2,\ldots,n$ and $j=1,2,\ldots,n$.

In this paper we are interested in finding matrix $D^{(n)}$, only. The comparison and computation of the cost is left for the host computer. In order to compute $D^{(n)}$ we use unidirectional linear systolic array (ULSA). Because of the problem dimension and data dependencies in the corresponding data dependency graph, Algorithm_1 is not suitable for direct synthesis of ULSA. To overcome this problem we partition the computations in Algorithm_1 into appropriate number of two-dimensional entities which are then computed on the ULSA The final result is obtained by repeating the computation n times on the designed ULSA. We require that designed ULSA is space-optimal with respect to a problem size and the execution time should be as minimal as possible for a given size of ULSA.

II. THE SYSTOLIC ALGORITHM

In order to obtain two-dimensional entities suitable for the synthesis of ULSA with desired properties, it is obvious that one of index variables in Algorithm_1 should be fixed on some constant value. The computation in Algorithm_1 does not depend on whether it is performed first on index variable i or j. In other words the loops on index variables i and j can be permuted. Therefore, without lost of generality, assume that index variable i is outer and fixed to some constant value. Two-dimensional entities obtained by setting i to some constant value in Algorithm_1 are $M_i=(d_{ij}^{(k)}), i=1,2,...,n$. The dependencies between different $M_i, i=1,2,...,n$ are very complex. Namely, it is not possible to compute M_i only according to $M_t, 0\le t\le i-1$. Therefore, the computation of different M_i is not suitable for systolic implementation. The same conclusion would be obtained if j is fixed. Therefore, only index variable k has left. Let us note that index variable k is an iterative one in Algorithm_1. The computations in SA

are usually performed such that resulting elements are pipelined through the array unless SA with fixed number of PEs is concerned or the array where resulting elements are accumulated in PEs. We do not use neither of the approaches in this paper. Instead, we partition the computations in Algorithm_1 into two-dimensional entities $D^{(k)} = (d_{ij}^{(k)})$, for some fixed k=1,2,...,n. The computation of $D^{(k)}$ depends only on $D^{(k-1)}$. This means that the computations of $D^{(1)}$,..., $D^{(n)}$ can be performed successively, which is very important for our approach. It is enough to synthesize ULSA that computes $D^{(1)}$ and then use it to compute $D^{(n)}$ by repeating the computations n times.

To ease the presentation we use the following denotation

$$a(i,0,1) = d_{i1}^{(0)}, b(0,j,1) = d_{ij}^{(0)}, c(i,j0) = d_{ij}^{(0)}, c(i,j,1) = d_{ij}^{(1)} \quad (1)$$

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

The inner computation space of Algorithm_2 is

$$P_{\text{int}} = \{(i,j,1) \mid 1 \le i \le n, 1 \le j \le n\} \quad (2)$$

and the corresponding dependency matrix is

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

Different linear SAs can be obtained by mapping (D, P_{int}) along different projection direction vectors. Namely, each projection direction vector 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 is of the form [9]:

$$T = \begin{bmatrix} \vec{\Pi} \\ S \end{bmatrix} = \begin{bmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ t_{31} & t_{32} & t_{33} \end{bmatrix}$$

where $\vec{\Pi}$ determines time scheduling, while S is space transformation which maps (D, P_{int}) into 1D systolic array. ULSA can be obtained for the direction projection vector $\vec{\mu} = [1 \quad -1 \quad 0]^T$ (see, for example [7-8]). However, this direction is not a permissible one for the Algorithm_2. Namely, if from the set of possible transformation matrices for this direction the following one

$$T = \begin{bmatrix} \vec{\Pi} \\ S \end{bmatrix} = \begin{bmatrix} 2 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (4)$$

is chosen arbitrarily, then according to mapping $S : D \to \Delta$, i.e. according to

$$\Delta = S \cdot D = \left[\vec{e}_b^{\,2} \quad \vec{e}_a^{\,2} \quad \vec{e}_c^{\,2}\right] = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (5)$$

which represents the direction of data flow in the systolic array, we conclude that $\vec{e}_b^{\,2} = \vec{e}_a^{\,2} = [1 \quad 0]^T$ meaning that elements of vectors $\vec{b}$ and $\vec{a}$ propagate through the array in the same direction. This will cause that the same partial product is computed in all PEs, leading to incorrect computation. One way to solve this problem is to introduce the delay elements between neighbouring PEs. This will result in different data speed of $\vec{b}$ and $\vec{a}$ elements through the array. If we put delay elements on $\vec{b}$ path, then $\vec{e}_b^{\,2}$ will be half of the $\vec{e}_a^{\,2}$ i.e. $\vec{e}_b^{\,2} = \frac{1}{2}\vec{e}_a^{\,2}$. From the algorithmic point of view, we need to introduce additional index space,

$$P_d = \{(i - \frac{1}{2}, j, 1) \mid 1 \le i \le n, 1 \le j \le n\} \quad (6)$$

and new dependency matrix

$$D = \left[\vec{e}_b^{\,3} \quad \vec{e}_a^{\,3} \quad \vec{e}_c^{\,3}\right] = \begin{bmatrix} \frac{1}{2} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (7)$$

Let us note that in index points of space P_{d} no computation is performed. Elements of vector $\vec{b}$ are just copied (i.e. delayed for one cycle). Now, according to (2), (6) and (7) we construct a new systolic algorithm, equivalent to Algorithm_2 which computes $D^{(1)}$, for which the direction $\vec{\mu} = [1 \quad -1 \quad 0]^T$ is a permissible one. The algorithm has the following form

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

The ULSA obtained according to Algorithm_3 computes $D^{(1)}$ correctly, but according to P_{int}, defined by (2) and space transformation S, defined by (4), it has $\Omega = 2n-1$ PEs which is too large for a given problem size. Space-optimal ULSA should have n PEs. In order to obtain ULSA with optimal number of PEs, the inner computation space P_{int} has to be accommodated to the projection direction $\vec{\mu} = [1 \quad -1 \quad 0]^T$ (see, for example, [7-9]). The accommodation is performed by mapping P_{int} into a new according to the following

$$\begin{bmatrix} U \\ V \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} i \\ j \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ n \\ 0 \end{bmatrix} = \begin{bmatrix} i \\ j - i + n \\ 1 \end{bmatrix}$$

for each i=1,2,...,n and j=1,2,...,n. Now, the new inner computation space is

$$P_{\text{int}} = \{(i, j-i+n,1) \mid 1 \le i \le n, 1 \le j \le n\}\ , \qquad (8)$$

and the corresponding space of delay elements is

$$P_d = \{(i-\frac{1}{2}, j-i+n,1) \mid 1 \le i \le n, 1 \le j \le n\}. \qquad (9)$$

Now, according to (7), (8) and (9), we can define the new systolic algorithm that is adjusted to direction $\vec{\mu} = [1 \quad -1 \quad 0]^T$ and equivalent to Algorithm_3. It has the following form

Algorithm_4
for i:=1 **to** n **do**
for j:=1 **to** n **do**

$$b(i-\frac{1}{2}, j-i+n,1) := b(i-1, j-i+n,1)$$
$$b(i, j-i+n,1) := b(i-\frac{1}{2}, j-i+n,1)$$
$$a(i, j-i+n,1) := a(i, j-i+n-1,1)$$
$$c(i, j-i+n,1) := \min\{c(i, j-i+n,0), \max\{a(i, j-i+n,1), b(i, j-i+n,1)\}\}$$

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

III. THE ULSA SYNTHESIS

The ULSA that computes matrix $D^{(1)}$ according to Algorithm_3 is obtained by mapping (P_{int}, P_{d}, D) using transformation S defined by (4). The (x,y) coordinates of the PEs in ULSA are obtained by mapping P_{int}, defined by (8), according to the following equations

$$PE \rightarrow \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} j+n \\ 1 \end{bmatrix} \qquad (10)$$

for each i=1,2,...,n.

Positions of delay elements in the (x,y) plane are determined by mapping set P_{d}, defined by (9), using transformation S defined by (4), i.e. according to

$$P_d \rightarrow \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} j+n-\frac{1}{2} \\ 1 \end{bmatrix} \qquad (11)$$

for each i=1,2,...,n.

The communication links between the PEs in the ULSA are implemented along the propagation vectors

$$\Delta = S \cdot D = \begin{bmatrix} \vec{e}_b^2 & \vec{e}_a^2 & \vec{e}_c^2 \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad (12)$$

where D is defined by (7) and S by (4).

The initial (x,y) positions of input data items at the beginning of the computation in the ULSA are obtained according to

$$a(i,01) \rightarrow \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1+n-i \\ 1 \end{bmatrix} + rn \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$
$$b(0, j-i+n,1) \rightarrow \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \frac{j-i+2n+1}{2} \\ 1 \end{bmatrix} + rn \begin{bmatrix} \frac{1}{2} \\ 0 \end{bmatrix}, \qquad (13)$$
$$c(i, j-i+n,0) \rightarrow \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} j+n \\ 2-i-j \end{bmatrix} + rn \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

for each i=1,2,...,n and j=1,2,..., n. The parameter r is determined for each pair (i, j) as greater of the integers from the set {0,1} for which the following is valid

$$2-i-j+rn \ge 0\ .$$

The role of parameter r is to minimize the execution time of Algorithm_4. An example of ULSA that computes $D^{(1)}$ for the case n=3 is depicted in Fig. 1. Denotation introduced in (1) is used.

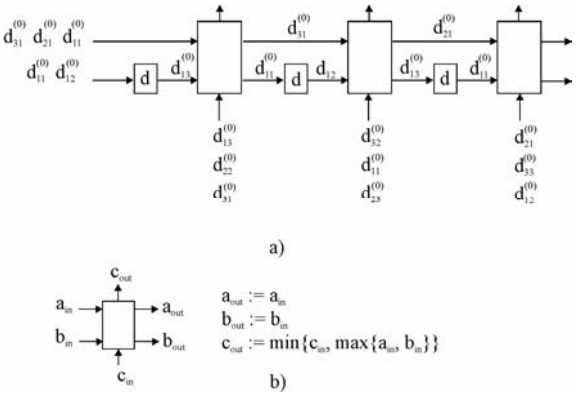

Fig.1. a) Data flow in the ULSA during the computation of $D^{(1)}$ for the case n=3. b) Functional property of the PE.

IV. PERFORMANCE ANALYSIS

According to (10) it is not difficult to conclude that obtained ULSA has $\Omega = n$ PEs, which is optimal number for a given problem size. Assume that time needed to perform an operation of type finding a minimum and maximum of two values (min and max) 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_4 on the ULSA. According to (13) we have that $t_{in}=2n-2$, $t_{exe} = n$ and $t_{out}=2n-2$. Since $t_{tot} = t_{in} + t_{exe} + t_{out}$ we have that $t_{tot}=5n-4$. Recall that we have designed ULSA that implements Algorithm_4, i.e. computes $D^{(1)} = (d_{ij}^{(1)})$. Matrix $D^{(n)}$ is obtained by computing $D^{(1)}, D^{(2)}, ..., D^{(n-1)}, D^{(n)}$, such that elements $D^{(k)} = (d_{ij}^{(k)})$ are used as inputs for computing $D^{(k+1)} = (d_{ij}^{(k+1)})$, for k=0,1,...,n-1. During the computation the output time of k-th iteration is overlapped with the input time of (k+1)-st iteration. This is illustrated in Table I for the case n=3. Having this in mind, the total time required to compute matrix $D^{(n)}$, on the ULSA is $T_{tot} = n(3n-2)$. The efficiency of the ULSA is $E = \frac{n}{3n-2}$, $0.33 \le E \le 0.5$, which is considered as good efficiency.

TABLE I
STEP BY STEP TIMMING DIAGRAM

clk	d	PE1	d	PE2	d	PE3
0	$d_{11}^{(0)}$					
1	$d_{13}^{(0)}$	$0 \coloneqq \min\{0, \max\{0, d_{11}^{(0)}\}\}$				
2	$d_{12}^{(0)}$	$0 \coloneqq \min\{0, \max\{0, d_{13}^{(0)}\}\}$	$d_{11}^{(0)}$			
3	$d_{11}^{(0)}$	$0 \coloneqq \min\{0, \max\{d_{21}^{(0)}, d_{12}^{(0)}\}\}$	$d_{13}^{(0)}$	$0 \coloneqq \min\{0, \max\{0, d_{11}^{(0)}\}\}$		
4	$d_{13}^{(0)}$	$0 \coloneqq \min\{0, \max\{d_{31}^{(0)}, d_{11}^{(0)}\}\}$	$d_{12}^{(0)}$	$0 \coloneqq \min\{0, \max\{d_{21}^{(0)}, d_{13}^{(0)}\}\}$	$d_{11}^{(0)}$	
5	$d_{12}^{(0)}$	$d_{13}^{(1)} \coloneqq \min\{d_{13}^{(0)}, \max\{d_{11}^{(0)}, d_{13}^{(0)}\}\}$	$d_{11}^{(0)}$	$d_{32}^{(1)} \coloneqq \min\{d_{32}^{(0)}, \max\{d_{31}^{(0)}, d_{12}^{(0)}\}\}$	$d_{13}^{(0)}$	$d_{21}^{(1)} \coloneqq \min\{d_{21}^{(0)}, \max\{d_{21}^{(0)}, d_{11}^{(0)}\}\}$
6	$d_{11}^{(0)}$	$d_{22}^{(1)} \coloneqq \min\{d_{22}^{(0)}, \max\{d_{21}^{(0)}, d_{12}^{(0)}\}\}$	$d_{13}^{(0)}$	$d_{11}^{(1)} \coloneqq \min\{d_{11}^{(0)}, \max\{d_{11}^{(0)}, d_{11}^{(0)}\}\}$	$d_{12}^{(0)}$	$d_{33}^{(1)} \coloneqq \min\{d_{33}^{(0)}, \max\{d_{31}^{(0)}, d_{13}^{(0)}\}\}$
7	$d_{11}^{(1)}$	$d_{31}^{(1)} \coloneqq \min\{d_{31}^{(0)}, \max\{d_{31}^{(0)}, d_{11}^{(0)}\}\}$	$d_{12}^{(0)}$	$d_{23}^{(1)} \coloneqq \min\{d_{23}^{(0)}, \max\{d_{21}^{(0)}, d_{13}^{(0)}\}\}$	$d_{11}^{(0)}$	$d_{12}^{(1)} \coloneqq \min\{d_{12}^{(0)}, \max\{d_{11}^{(0)}, d_{12}^{(0)}\}\}$
8	$d_{13}^{(1)}$	$0 \coloneqq \min\{0, \max\{0, d_{11}^{(1)}\}\}$	$d_{11}^{(0)}$	$0 \coloneqq \min\{0, \max\{d_{31}^{(0)}, d_{12}^{(0)}\}\}$	$d_{13}^{(0)}$	$0 \coloneqq \min\{0, \max\{d_{21}^{(0)}, d_{11}^{(0)}\}\}$
9	$d_{12}^{(1)}$	$0 \coloneqq \min\{0, \max\{0, d_{13}^{(1)}\}\}$	$d_{11}^{(1)}$	$0 \coloneqq \min\{0, \max\{0, d_{11}^{(0)}\}\}$	$d_{12}^{(0)}$	$0 \coloneqq \min\{0, \max\{d_{31}^{(0)}, d_{13}^{(0)}\}\}$
10	$d_{11}^{(1)}$	$0 \coloneqq \min\{0, \max\{d_{21}^{(1)}, d_{12}^{(1)}\}\}$	$d_{13}^{(1)}$	$0 \coloneqq \min\{0, \max\{0, d_{11}^{(1)}\}\}$	$d_{11}^{(0)}$	$0 \coloneqq \min\{0, \max\{0, d_{12}^{(0)}\}\}$
11	$d_{13}^{(1)}$	$0 \coloneqq \min\{0, \max\{d_{31}^{(1)}, d_{11}^{(1)}\}\}$	$d_{12}^{(1)}$	$0 \coloneqq \min\{0, \max\{d_{21}^{(1)}, d_{13}^{(1)}\}\}$	$d_{11}^{(1)}$	$0 \coloneqq \min\{0, \max\{0, d_{11}^{(0)}\}\}$
12	$d_{12}^{(2)}$	$d_{13}^{(2)} \coloneqq \min\{d_{13}^{(1)}, \max\{d_{11}^{(1)}, d_{13}^{(1)}\}\}$	$d_{11}^{(1)}$	$d_{32}^{(2)} \coloneqq \min\{d_{32}^{(1)}, \max\{d_{31}^{(1)}, d_{12}^{(1)}\}\}$	$d_{13}^{(1)}$	$d_{21}^{(2)} \coloneqq \min\{d_{21}^{(1)}, \max\{d_{21}^{(1)}, d_{11}^{(1)}\}\}$
13	$d_{11}^{(1)}$	$d_{22}^{(2)} \coloneqq \min\{d_{22}^{(1)}, \max\{d_{21}^{(1)}, d_{12}^{(1)}\}\}$	$d_{13}^{(1)}$	$d_{11}^{(2)} \coloneqq \min\{d_{11}^{(1)}, \max\{d_{11}^{(1)}, d_{11}^{(1)}\}\}$	$d_{12}^{(1)}$	$d_{33}^{(2)} \coloneqq \min\{d_{33}^{(1)}, \max\{d_{31}^{(1)}, d_{13}^{(1)}\}\}$
14	$d_{11}^{(2)}$	$d_{31}^{(2)} \coloneqq \min\{d_{31}^{(1)}, \max\{d_{31}^{(1)}, d_{11}^{(1)}\}\}$	$d_{12}^{(1)}$	$d_{23}^{(2)} \coloneqq \min\{d_{23}^{(1)}, \max\{d_{21}^{(1)}, d_{13}^{(1)}\}\}$	$d_{11}^{(1)}$	$d_{12}^{(2)} \coloneqq \min\{d_{12}^{(1)}, \max\{d_{11}^{(1)}, d_{12}^{(1)}\}\}$
15	$d_{13}^{(2)}$	$0 \coloneqq \min\{0, \max\{0, d_{11}^{(2)}\}\}$	$d_{11}^{(1)}$	$0 \coloneqq \min\{0, \max\{d_{31}^{(1)}, d_{12}^{(1)}\}\}$	$d_{13}^{(1)}$	$0 \coloneqq \min\{0, \max\{d_{21}^{(1)}, d_{11}^{(1)}\}\}$
16	$d_{12}^{(2)}$	$0 \coloneqq \min\{0, \max\{0, d_{13}^{(2)}\}\}$	$d_{11}^{(2)}$	$0 \coloneqq \min\{0, \max\{0, d_{11}^{(1)}\}\}$	$d_{12}^{(1)}$	$0 \coloneqq \min\{0, \max\{d_{31}^{(1)}, d_{13}^{(1)}\}\}$
17	$d_{11}^{(2)}$	$0 \coloneqq \min\{0, \max\{d_{21}^{(2)}, d_{12}^{(2)}\}\}$	$d_{13}^{(2)}$	$0 \coloneqq \min\{0, \max\{0, d_{11}^{(2)}\}\}$	$d_{11}^{(1)}$	$0 \coloneqq \min\{0, \max\{0, d_{12}^{(1)}\}\}$
18	$d_{13}^{(2)}$	$0 \coloneqq \min\{0, \max\{d_{31}^{(2)}, d_{11}^{(2)}\}\}$	$d_{12}^{(2)}$	$0 \coloneqq \min\{0, \max\{d_{21}^{(2)}, d_{13}^{(2)}\}\}$	$d_{11}^{(2)}$	$0 \coloneqq \min\{0, \max\{0, d_{11}^{(1)}\}\}$
19	$d_{12}^{(2)}$	$d_{13}^{(3)} \coloneqq \min\{d_{13}^{(2)}, \max\{d_{11}^{(2)}, d_{13}^{(2)}\}\}$	$d_{11}^{(2)}$	$d_{32}^{(3)} \coloneqq \min\{d_{32}^{(2)}, \max\{d_{31}^{(2)}, d_{12}^{(2)}\}\}$	$d_{13}^{(2)}$	$d_{21}^{(3)} \coloneqq \min\{d_{21}^{(2)}, \max\{d_{21}^{(2)}, d_{11}^{(2)}\}\}$
20	$d_{11}^{(2)}$	$d_{22}^{(3)} \coloneqq \min\{d_{22}^{(2)}, \max\{d_{21}^{(2)}, d_{12}^{(2)}\}\}$	$d_{13}^{(2)}$	$d_{11}^{(3)} \coloneqq \min\{d_{11}^{(2)}, \max\{d_{11}^{(2)}, d_{11}^{(2)}\}\}$	$d_{12}^{(2)}$	$d_{33}^{(3)} \coloneqq \min\{d_{33}^{(2)}, \max\{d_{31}^{(2)}, d_{13}^{(2)}\}\}$
21		$d_{31}^{(3)} \coloneqq \min\{d_{31}^{(2)}, \max\{d_{31}^{(2)}, d_{11}^{(2)}\}\}$	$d_{12}^{(2)}$	$d_{23}^{(3)} \coloneqq \min\{d_{23}^{(2)}, \max\{d_{21}^{(2)}, d_{13}^{(2)}\}\}$	$d_{11}^{(2)}$	$d_{12}^{(3)} \coloneqq \min\{d_{12}^{(2)}, \max\{d_{11}^{(2)}, d_{12}^{(2)}\}\}$
22						

V. CONCLUSION

A problem of finding MCST of a given graph was considered. The problem is partitioned into two parts. First, we compute a wighted matrix, $D^{(n)}$, according to weighted matrix, $D^{(0)}$, of a given graph. Second, MCST is determined by comparing matrices $D^{(n)}$ and $D^{(0)}$. To solve the first problem we designed unidirectional linear systolic array with optimal number of PEs and minimize the execution time. The efficiency of the designed array is in the range [0.33, 0.5]. Second task is performed by the host computer.

REFERENCES

[1] O. Boruvka, "About a certain minimal problem", *Prace Mor.Prirodoved.Spol.v Brne,* vol. III, no. 3, pp. 37-58, 1926.

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

[3] R. C. Prim, "The shortest connecting network and some generalisations", *Bell Syst. Tech. J.*, no.36, pp. 1389-1401, 1957.

[4] J. B. Kruskal, " On the shortest spanning subtree of a graph and the travelling salesman problem", *Proc. Amer. Math. Soc.*, no 7, pp. 48-50, 1956.

[5] S. Warshall. "A theorem on Boolean matrices", *J. ACM*, no. 9, pp. 11-12, 1962.

[6] R. W. Floyd,"Algorithm 97, Shortest path", C. ACM, no. 5, pp. 5, 1962.

[7] E. I. Milovanović, I. Ž. Milovanović, B. M. Randjelović, M. K. Stojčev, "Systolic implementation of nonlinear transformation of two sequences", *TELSIKS '03 Conference Proceedings*, pp.592-595, Nis, Serbia & Montenegro, 2003.

[8] E. I. Milovanović, M. P. Bekakos, Ć. Dolićanin, I. Ž. Milovanović, "Computing transitive closure on unidirectional linear systolic array, *J. Electrotehn. Math*., vol. 9, no 1, pp.19-28, 2004.

[9] M. Bekakos, E. Milovanović, N. Stojanović, T. Tokić, I. Milovanović, I. Milentijević, "Transformation matrices for systolic array synthesis", *J. Electrotehn. Math*., vol. 7, no 1, pp.9-15, 2002.