Fault-Tolerant Systolic Array for Matrix Multiplication

E. I. Milovanović, I.Ž. Milovanović, N. M. Stojanović, T. I. Tokić, M. K. Stojčev

Abstract— **An approach to design fault-tolerant hexagonal systolic array (SA) for multiplication of rectangular matrices is described. Fault tolerance is achieved through triplicated computation of the same problem instance and majority voting.**

Keywords— **Matrix multiplication, fault-tolerance, systolic arrays**

I. Introduction

Fault tolerance has become a crucial design requirement for VLSI/WSI array processors. Fault tolerance can be achieved through some form of redundancy, i.e. either information (ABFT) [1], [2], space and/or time [3], [4], and by reconfiguration. In this paper we present a systematic approach to design a fault–tolerant VLSI/WSI SA for matrix multiplication. The method is based on fault–tolerant (FT) mapping theory which is developed from space–time mapping technique [5], [6], [7], and the theory on concurrent error detection using space/time redundancy [4], [8]. Fault tolerance is achieved through triplicated computation of the same problem instance and majority voting. Our mapping algorithm to obtain fault–tolerant systolic array is based on composition of two linear transformations (H, T), contrary to the approach proposed in [4], which uses only valid transformation matrix T. The total number of PEs, n_p, and the total computation time, t_c, are two major performance measures. Therefore we take their product $AT = n_p \times t_c$ as the indicator of the SA optimality.

II. Background

Let $A = (a_{ik})$ and $B = (b_{kj})$ be two matrices of order $N_1 \times N_3$ and $N_3 \times N_2$, respectively. In order to design an optimal fault–tolerant hexagonal SA for computing $C = A \times B$, three equivalent algorithms, but with disjoint index spaces were proposed in [4]. The main idea in [4] was to achieve fault–tolerance through repeated computation of the same problem instance. By the proposed method any single error at any given time can be detected and corrected. The common description of all three algorithms is as follows

Algorithms_1
for $r := 0$ **to** 2 **do**
for $k := 1$ **to** N_3 **do**
for $j := 1$ **to** N_2 **do**
for $i := 1$ **to** N_1 **do**
$a(i-\frac{2r}{3}, j+\frac{r}{3}, k+\frac{r}{3}) := a(i-\frac{2r}{3}, j+\frac{r}{3}-1, k+\frac{r}{3})$;
$b(i-\frac{2r}{3}, j+\frac{r}{3}, k+\frac{r}{3}) := b(i-\frac{2r}{3}-1, j+\frac{r}{3}, k+\frac{r}{3})$;
$c(i-\frac{2r}{3}, j+\frac{r}{3}, k+\frac{r}{3}) := c(i-\frac{2r}{3}, j+\frac{r}{3}, k+\frac{r}{3}-1)+$
$a(i-\frac{2r}{3}, j+\frac{r}{3}, k+\frac{r}{3}) * b(i-\frac{2r}{3}, j+\frac{r}{3}, k+\frac{r}{3})$;
endfor$\{i, j, k, r\}$;

For $r = 0, 1, 2$ three equivalent algorithms with disjoint index spaces are obtained. The initial values in Algorithms_1 are given by $a(i-\frac{2r}{3}, \frac{r}{3}, k+\frac{r}{3}) \equiv a_{ik}$, $b(-\frac{2r}{3}, j+\frac{r}{3}, k+\frac{r}{3}) \equiv b_{kj}$, and $c(i-\frac{2r}{3}, j+\frac{r}{3}, \frac{r}{3}) \equiv 0$. The output values of Algorithms_1 are $c(i-\frac{2r}{3}, j+\frac{r}{3}, N_3+\frac{r}{3}) \equiv c_{ij}$.

The computational structures of the above matrix multiplication algorithms are determined by the inner computation spaces

$$P_{int}(r)=\{(i-\tfrac{2r}{3}, j+\tfrac{r}{3}, k+\tfrac{r}{3}) | 0 \le r \le 2, \\ 1 \le i \le N_1, 1 \le j \le N_2, 1 \le k \le N_3\}$$

where data are used or computed, and a dependency matrix

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

Using valid transformation matrix

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

the computational structure $(D, P_{int}(r))$ is mapped into fault–tolerant SA, i.e.

$$T : (D, P_{int}(r)) \longmapsto (\Delta, \bar{P}_{int}(r)), \quad (3)$$

where $\bar{P}_{int}(r) = \{(t,\ x,\ y)\}$. Here t represents the time when computation is performed, while (x, y) denotes the coordinates of the PE were the computation is taking place. Both refer to the index point (i, j, k). Note that several valid transformations T that map $(D, P_{int}(r))$ into $(\Delta, \bar{P}_{int}(r))$, can be generated. For T defined by (2), the (x, y) positions of PEs in the SA are given by

$$\begin{bmatrix} x \\ y \end{bmatrix} = S \cdot \vec{p} = \begin{bmatrix} j-i+r \\ k-j \end{bmatrix}, \quad (\vec{p} \in P_{int}(r)) \quad (4)$$

I.Ž. Milovanović, E. I. Milovanović, T. I. Tokić, N. M. Stojanović, M. K. Stojčev are with the Faculty of Electronic Engineering, University of Niš, Beogradska 14, 18000 Niš, Serbia, E-mail: ema@elfak.ni.ac.yu

while time schedule, t, is determined according to

$$t = t(i - \frac{2r}{3}, j + \frac{r}{3}, k + \frac{r}{3}) = i + j + k. \qquad (5)$$

The communication links between the PEs are implemented along the projections of data dependency vectors, i.e.

$$\Delta_s = [\vec{e}_b^{\,2} \vec{e}_a^{\,2} \vec{e}_c^{\,2}] = S \cdot D = \begin{bmatrix} -1 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix}. \qquad (6)$$

The array obtained according to (4), (5) and (6) has the following features:

- number of PEs: $n_p = N_1(N_2 - 1) + N_2(N_3 + 1) + N_3(N_1 + 1) - 1$,
- computation time: $t_c = N_1 + N_2 + N_3 - 2$,
- AT factor: $AT = n_p \times t_c = [N_1(N_2 - 1) + N_2(N_3 + 1) + N_3(N_1 + 1) - 1][N_1 + N_2 + N_3 - 2]$.

For the case $N_1 = N_2 = N_3 = N$ the following is obtained [4]:

$$\begin{aligned} n_p &= 3N^2 + N - 1, \qquad t_c = 3N - 2, \\ AT &= (3N^2 + N - 1)(3N - 2) = O(9N^3) \qquad (7) \end{aligned}$$

The corresponding SA that implements fault–tolerant matrix multiplication for $N_1 = N_2 = N_3 = 3$ is shown in Fig.1. This array is referred to as optimal fault-tolerant SA with respect to AT factor in [4].

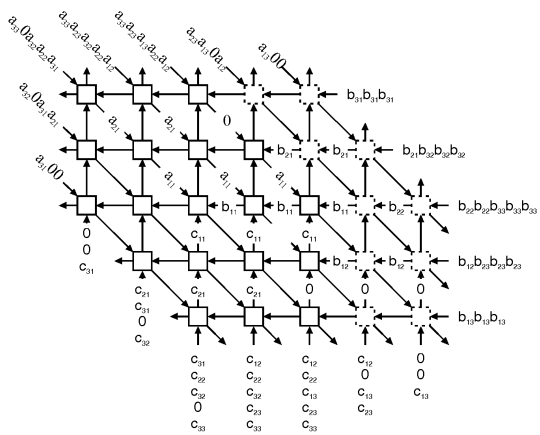

Fig. 1. Data flow in the SA during fault-tolerant matrix multiplication for $N_1 = N_2 = N_3 = 3$, obtained in [4].

III. Our approach to mapping matrix multiplication algorithm onto FT SA

This section describes a modification of the method given in [4] to synthesize a hexagonal SA which implements fault–tolerant matrix multiplication with minimal number of PEs with respect to the problem size. To achieve this, we apply some linear mappings that accommodate the inner computation space $P_{int}(r)$ to the projection direction $\vec{\mu} = [1\,1\,1]^T$ prior to the mapping defined by (3). The accommodation of $P_{int}(r)$ can be performed both on index variable i and j. The choice depends on the relation between N_1 and N_2 (i.e. $N_1 > N_2$ or $N_1 < N_2$). When $N_1 = N_2$ both accommodations are equally profitable.

We will first consider the case when $N_1 > N_2$, i.e. when accommodation is performed on the index variable i. In this case, instead of Algorithms_1, we consider

Algorithms_2
for $r := 0$ **to** 2 **do**
 for $k := 1$ **to** N_3 **do**
  for $j := 1$ **to** N_2 **do**
   for $i := 1$ **to** N_1 **do**
   $a(i + \frac{r}{3}, j - r, k) := a(i + \frac{r}{3}, j - r - 1, k)$;
   $b(i + \frac{r}{3}, j - r, k) := b(i + \frac{r}{3} - 1, j - r, k)$;
   $c(i + \frac{r}{3}, j - r, k) := c(i + \frac{r}{3}, j - r, k - 1) +$
   $a(i + \frac{r}{3}, j - r, k) * b(i + \frac{r}{3}, j - r, k)$;
endfor$\{i, j, k, r\}$;

with the following initial values $a(i + \frac{r}{3}, -r, k) \equiv a_{ik}$, $b(\frac{r}{3}, j - r, k) \equiv b_{kj}$, and $c(i + \frac{r}{3}, j - r, 0) \equiv 0$. The output values of Algorithms_2 are $c(i + \frac{r}{3}, j - r, N_3) \equiv c_{ij}$. Note that computations in Algorithms_2 satisfy all the conditions given in [4] to perform fault–tolerant matrix multiplication. The inner computation spaces of Algorithms_2 are given by

$$\begin{aligned} P_{int}(r) &= \{(i + \tfrac{r}{3}, j - r, k) | 0 \le r \le 2,\ 1 \le i \le N_1, \\ &\quad 1 \le j \le N_2\ 1 \le k \le N_3\}. \end{aligned}$$

The data dependency matrix is the same as for Algorithms_1, i.e. as one given by (1).

The accommodation of $P_{int}(r)$ to the direction $\vec{\mu} = [1\,1\,1]^T$ over index variable i is performed by linear mapping $H = (F, G)$ (see for example [6], [7]) according to

$$H : P_{int}(r) \longmapsto P^*_{int}(r), \quad \text{i.e.} \quad \vec{p}^{\,*} = F \cdot \vec{p} + G \qquad (8)$$

for each $\vec{p} \in P_{int}(r)$, where

$$F = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}, \quad \text{and } G = \begin{bmatrix} 0 \\ -1 \\ -1 \end{bmatrix}. \qquad (9)$$

According to (8) the elements $\vec{p}^{\,*} = [u\,v\,w]^T$ of index space $P^*_{int}(r)$ are obtained as

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = F \cdot \begin{bmatrix} i + \frac{r}{3} \\ j - r \\ k \end{bmatrix} + G = \begin{bmatrix} i + \frac{r}{3} \\ i + j - \frac{2r}{3} - 1 \\ i + k + \frac{r}{3} - 1 \end{bmatrix}, \qquad (10)$$

for each $0 \le r \le 2$, $1 \le i \le N_1$, $1 \le j \le N_2$, $1 \le k \le N_3$. The SA that implements fault–tolerant matrix multiplication is obtained by the following mapping

$$T : (D, P^*_{int}(r)) \longmapsto (\Delta, \bar{P}_{int}(r)), \qquad (11)$$

where the transformation T is given by

$$T = \begin{bmatrix} \vec{\Pi} \\ S \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ -- & -- & -- \\ -1 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix} \qquad (12)$$

Note that transformation matrix T is not the same as one given by (2). Namely, under the conditions defined in [6], the transformation T given by (2) is no more valid, since it would generate SA with bad spatial features.

The (x, y) coordinates of the PEs in the SA are determined from

$$\begin{bmatrix} x \\ y \end{bmatrix} = S \cdot \vec{p}^* = \begin{bmatrix} j - r - 1 \\ k - 1 \end{bmatrix},$$

for each $0 \le r \le 2$, $1 \le i \le N_1$, $1 \le j \le N_2$, $1 \le k \le N_3$. Time schedule of the data item indexed by (i, j, k) is given by

$$t = t(i + \frac{r}{3}, i + j - \frac{2r}{3} - 1, i + k + \frac{r}{3} - 1) = 3i + j + k - 2.$$

Note that for input data items we assume the following periodicity

$$a(i + \frac{r}{3} + N_1, j - \frac{2r}{3}, k + \frac{r}{3}) \equiv a(i + \frac{r}{3}, j - \frac{2r}{3}, k + \frac{r}{3} + N_3) \equiv a_{ik}$$

$$b(i + \frac{r}{3}, j - \frac{2r}{3} + N_2, k + \frac{r}{3}) \equiv b(i + \frac{r}{3}, j - \frac{2r}{3}, k + \frac{r}{3} + N_3) \equiv b_{kj}.$$

The initial (x, y) positions of input data items are determined from

$$a(i + \frac{r}{3}, 0, i + k + \frac{r}{3} - 1) \mapsto \begin{bmatrix} 3 - 3i - k - r \\ k - 1 \end{bmatrix}$$

$$b(0, i + j - 1 - \frac{2r}{3}, i + k + \frac{r}{3} - 1) \mapsto \begin{bmatrix} 3i + 2j + k - r - 5 \\ 3i + j + 2k - 5 \end{bmatrix}$$

$$c(i + \frac{r}{3}, i + j - \frac{2r}{3} - 1, 0) \mapsto \begin{bmatrix} j - r - 1 \\ 3 - 3i - j \end{bmatrix},$$

for each $0 \le r \le 2$, $1 \le i \le N_1$, $1 \le j \le N_2$, $1 \le k \le N_3$. The obtained SA has the following features

$$n_p = N_3(N_2 + 2), \quad t_c = 3N_1 + N_2 + N_3 - 4,$$
$$AT = N_3(N_2 + 2)(3N_1 + N_2 + N_3 - 4) \qquad (13)$$

For $N_1 = N_2 = N_3 = N$ we have

$$n_p = N(N + 2), \quad t_c = 5N - 4,$$
$$AT = N(N + 2)(5N - 4) = O(5N^3). \qquad (14)$$

When $N_1 < N_2$ accommodation of $P_{int}(r)$ is performed over index variable j. Now, we start from the following algorithm

Algorithms_3
for $r := 0$ **to** 2 **do**
 for $k := 1$ **to** N_3 **do**
  for $j := 1$ **to** N_2 **do**
   for $i := 1$ **to** N_1 **do**
   $a(i - r, j + \frac{r}{3}, k) := a(i - r, j + \frac{r}{3} - 1, k)$;
   $b(i - r, j + \frac{r}{3}, k) := b(i - r - 1, j + \frac{r}{3}, k)$;
   $c(i - r, j + \frac{r}{3}, k) := c(i - r, j + \frac{r}{3}, k - 1) +$
   $a(i - r, j + \frac{r}{3}, k) * b(i - r, j + \frac{r}{3}, k)$;
endfor$\{i, j, k, r\}$;

with the following initial values $a(i - r, \frac{r}{3}, k) \equiv a_{ik}$, $b(-r, j + \frac{r}{3}, k) \equiv b_{kj}$, and $c(i - r, j + \frac{r}{3}, 0) = 0$. The output values are $c(i - r, j + \frac{r}{3}, k + N_3) \equiv c_{ij}$. The inner computation spaces of Algorithms_3 are given by

$$P_{int}(r) = \{(i - r, j + \tfrac{r}{3}, k) | 0 \le r \le 2,\ 1 \le i \le N_1,\ 1 \le j \le N_2\ 1 \le k \le N_3\}.$$

Data dependency matrix is the same as one given by (1). The accommodation of $P_{int}(r)$ to the direction $\vec{\mu} = [1\,1\,1]^T$ over index variable j, is performed by linear mapping of type (8), with $H = (F, G)$ being defined as [6], [7]:

$$F = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}, \text{ and } G = \begin{bmatrix} -1 \\ 0 \\ -1 \end{bmatrix}, \qquad (15)$$

The elements $\vec{p}^* = [u, v, w] \in P^*_{int}(r)$ are determined according to

$$\vec{p}^* = \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} i + j - \frac{2r}{3} - 1 \\ j + \frac{r}{3} \\ j + k + \frac{r}{3} - 1 \end{bmatrix},$$

for each $0 \le r \le 2$, $1 \le i \le N_1$, $1 \le j \le N_2$, $1 \le k \le N_3$. It is not hard to see that for mapping $H = (F, G)$ defined by (15), the transformation T given by (12) is no more appropriate one (see for example [6]). According to the criteria established in [6], for T we can choose

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

Now, the (x, y) positions of the PEs in the SA are determined from

$$\begin{bmatrix} x \\ y \end{bmatrix} = S \cdot \vec{p}^* == \begin{bmatrix} i - r - 1 \\ 1 - k \end{bmatrix}$$

for each $0 \le r \le 2$, $1 \le i \le N_1$, $1 \le j \le N_2$, $1 \le k \le N_3$. Time schedule of the data item indexed by (i, j, k) is given by

$$t = t(i + j - \frac{2r}{3} - 1, j + \frac{r}{3}, j + k + \tfrac{r}{3} - 1) = i + 3j + k - 2.$$

Note that for input data items we assume the following periodicity

$$a(i - \frac{2r}{3} + N_1, j + \frac{r}{3}, k + \frac{r}{3}) \equiv a(i - \frac{2r}{3}, j + \frac{r}{3}, k + \frac{r}{3} + N_3) \equiv a_{ik}$$

$$b(i - \frac{2r}{3}, j + \frac{r}{3} + N_2, k + \frac{r}{3}) \equiv b(i - \frac{2r}{3}, j + \frac{r}{3}, k + \frac{r}{3} + N_3) \equiv b_{kj}.$$

The initial (x, y) positions of input data items are determined from

$$a(i + j - \frac{2r}{3} - 1, 0, j + k + \frac{r}{3} - 1) \mapsto \begin{bmatrix} 2i + 3j + k - r - 5 \\ 5 - 3j - i - 2k \end{bmatrix}$$

$$b(0, j + \frac{r}{3}, j + k + \frac{r}{3} - 1) \mapsto \begin{bmatrix} 3 - 3j - k - r \\ 1 - k \end{bmatrix}$$

$$c(i + j - \frac{2r}{3} - 1, j + \frac{r}{3}, 0) \mapsto \begin{bmatrix} i - r - 1 \\ 3j + i - 3 \end{bmatrix},$$

for each $0 \le r \le 2$, $1 \le i \le N_1$, $1 \le j \le N_2$, $1 \le k \le N_3$.

The obtained array has the following features

$$n_p = N_3(N_1+2), \qquad t_c = N_1 + 3N_2 + N_3 - 4,$$
$$AT = N_3(N_1+2)(N_1+3N_2+N_3-4).$$

As we have already mentioned depending on the relation between N_1 and N_2, we can choose between Algorithms_2 and Algorithms_3 as a starting point for the synthesis procedure. Accordingly, we conclude that a hexagonal SA for fault–tolerant matrix multiplication has the following characteristics

$$n_p = N_3(\min\{N_1, N_2\} + 2),$$
$$t_c = (3\max\{N_1, N_2\} + \min\{N_1, N_2\} + N_3 - 4)$$
$$AT = N_3(\min\{N_1, N_2\} + 2)(3\max\{N_1, N_2\}$$
$$+ \min\{N_1, N_2\} + N_3 - 4).$$

For a given problem size this array has minimal possible number of PEs needed to perform fault–tolerant matrix multiplication. Compared to the array obtained in [4], the number of PEs is reduced almost three times, AT measure two times, while the total execution time is only slightly increased. Data flow in this array during fault–tolerant matrix multiplication for $N_1 = N_2 = N_3 = 3$ is diagrammed in Fig2. A detail concerning the voting mechanism is sketched in Fig.3. Figure 3. depicts only the processing elements at the boundary of fault-tolerant SA, since only the final results are subjects in the voting process. Note that each multiplexer (MUX) takes data from three different PEs at three different cycles. The inputs are selected in "round robin" manner starting from input 0. Control signals for all multiplexers are unique. Each voter takes three results to vote. There are $[\frac{N+2}{3}] * 3$ multiplexers and $[\frac{N+2}{3}]$ voters. By the proposed scheme a single permanent or temporary faults can be tolerated. A number of multiple fault patterns can be tolerated also, provided that faults do not affect the same element of the resulting matrix. Fault detection and location are not necessary for fault-tolerance, errors are masked concurrently with normal operation of the systolic array.

IV. Conclusion

We have described a method to synthesize optimal fault–tolerant SA for matrix multiplication with minimal hardware overhead. The array is optimal in the sense of the product of computation time and number of PEs required. The fault tolerance is achieved through triplicated computation of the same problem instance followed by the majority voting. A single permanent and temporary faults and a number of multiple fault patterns can be tolerated by the proposed scheme.

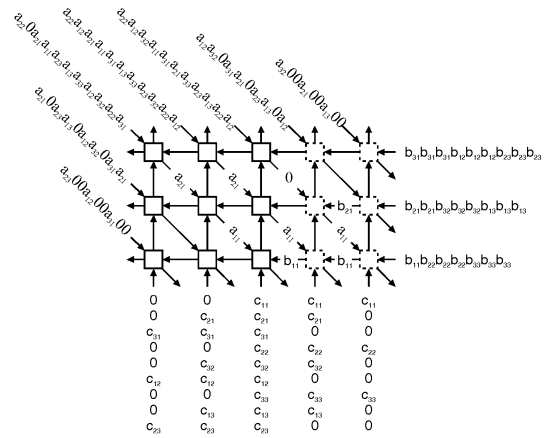

Fig. 2. Data flow in the SA synthesized by the described procedure during fault–tolerant matrix multiplication for $N_1 = N_2 = N_3 = 3$.

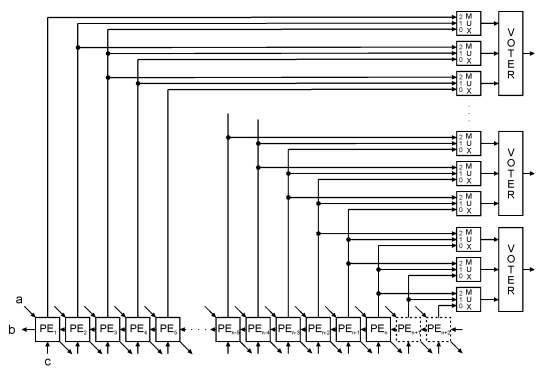

Fig. 3. A detail of the voting process.

References

[1] K. H. Huang,J. A. Abraham, *Algorithm–based fault tolerance for matrix operations*, IEEE Trans. Comput.,Vol. C–33, 6(1984), 518–528.

[2] M.K. Stojčev, E.I. Milovanović, I.Ž. Milovanović, *Algorithm based fault–tolerant technique for matrix inversion*, Proc. of the Int. Conf. Parallel Computing Technologies (ed. N. N. Mirenkov) Novosibirsk'91, World Scientific, 1991, 375–384.

[3] M. O. Esonu, A. J. Al–Khalili, S. Hariri, D. Al-Khalili, *Fault–tolerant design methodology for systolic array architectures*, IEE Proc. Comput. Digit. Tech., Vol. 141, 1 (1994), 17–28.

[4] C. N. Zhang, T. M. Bachtiar, W. K. Chou, *Optimal fault–tolerant design approach for VLSI array processors*, IEE Proc. Comput. Digit. Tech., Vol. 144, 1 (1997), 15–21

[5] I. Z. Milentijević, I. Ž. Milovanović, E. I. Milovanović, M. K. Stojčev, *The design of optimal planar systolic arrays for matrix multiplication*, Computers Math. Applic. Vol. 33, 6 (1997), 17–35.

[6] 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'98 (U.Gudukbay, Ed.), IOS Press, 1998, 389-396.

[7] I.Ž. Milovanović, T.I. Tokić, M.K. Stojčev, E.I. Milovanović, N.M. Novaković, *Mapping matrix multiplication algorithm onto optimal fault-tolerant systolic array*, Proc: 22nd Inter. Confer. Microelectronics, Niš'2000, Vol. 2 (2000), 711-714.

[8] J. -J. Wang, C.-W. Jen, *Redundancy design for a fault tolerant systolic array*, IEE Proceedings, Vol. 137. Pt. E, 3 (1990), 218-226.