FAULT-TOLERANT MATRIX MULTIPLICATION SYSTOLIC ARRAY

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

Faculty of Electronic Engineering

Beogradska 14, P.O. Box 73, 18000 Niš, Serbia

Abstract: An approach to design fault tolerant hexagonal systolic array (SA) for multiplication of rectangular matrices is described. The approach comprises of three steps. First, redundancies are introduced at the computational level by deriving three equivalent algorithms but with disjoint index spaces. Second, we perform the accommodation of index spaces to the projection direction to obtain hexagonal SA with optimal number of processing elements (PE) for a given problem size. Finally, we perform mapping of the accommodated index spaces into fault tolerant systolic array using valid transformation matrix. As a result we obtain SA with optimal number of PEs which perform fault-tolerant matrix multiplication. In the case of square matrices of order $N \times N$ this array comprises of $N^2 + 2N$ PEs with active computation time $t_c = 5N - 4$ time units.

Keyword: Matrix multiplication, fault-tolerance, systolic arrays

1. Introduction

Matrix multiplication plays a central role in numerical linear algebra, since we met it in almost all numerical algorithms, including matrix inversion, eigenvalue computation, numerical solution of the PDE, as well as in many technical problems including digital signal processing, circuit simulation, digital control, etc. Matrix multiplication is a very regular computation and lends itself well to parallel implementation. Regular structures such as systolic arrays (SA), are well suited for matrix multiplication and are also amenable to VLSI/WSI implementation because of simple and regular design, and nearest neighbor communications.

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 [3-5], and by reconfiguration.

In this paper we present a systematic approach to design a fault--tolerant VLSI/WSI SA for multiplication of matrix A of order $N_1 \times N_3$ by matrix B of order $N_3 \times N_2$ to obtain matrix C of order $N_1 \times N_2$. The method is based on fault-tolerant mapping theory which is developed from space-time mapping technique

[6-10], and the theory on concurrent error detection using space/time redundancy [4,5,11]. 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 [5], which uses only valid transformation matrix T. Linear transformation H accommodates index space of the matrix multiplication algorithm to the projection direction μ. This accommodation enables us to obtain SA with optimal number of processing elements (PE). Optimal SA is obtained when valid transformation T is applied on the accommodated index space. The total number of Pes, np, and the total computation time, tc, are two major performance measures. Therefore we take their product AT = np×tc as the indicator of the SA optimality. The fault tolerant SA for matrix multiplication obtained by the proposed method has $AT = N(\min\{N_1, N_2\} + 2)(3\max\{N_1, N_2\} + \min\{N_1, N_2\} + N_3 - 4)$ compared to AT = (N1(N2 – 1) N + N2 (N3 + 1) + N3(N1+1) – 1) (N1 + N2 + N3 – 2) obtained in [5]. When N1 = N2 = N3 = N we respectively obtain AT = (N2 + 2N) + (5N – 4) = O (5N3), compared to AT = (3N2 + N – 1) (3N – 2) = O (9N3).

2. Background

An important performance parameter of the designed SA is the pipeline period α which is the time interval in clock units between two successive activities of a processor. $\alpha = 1$ means that a processor is busy in every clock, $\alpha = 2$ means the processor is activated in every other clock, etc. A systolic array is not fully utilized if $\alpha > 1$. One way to increase the SA utilization is to execute two or more problem instances simultaneously (see for example [10]). Another is to perform replicated computations of the same problem instance to achieve fault tolerance (see for example [3, 5, 11]).

In error masking approach, which is known as N-tuple modular redundancy, N copies (N odd) of a module and majority voter are used to mask the error from failed module. At least three modules are necessary in a voting system which is typically called triple modular redundancy (TMR). It seams that at least 200 percent hardware overhead for fault tolerance is needed. In practice it needs to put triplicate computations to the voter and then gets a correct result. The triplicated computations may be computed in different PEs and/ or different time using space-shift, time-shift or space-time-shift [11]. If the replicated computations are performed simultaneously by different PEs it is a space-shift scheme. If the replicated computations are computed by the same PE at different times it is a time-shift scheme. If the replicated computations are computed by the same PE at dif-

ferent times it is a time-shift scheme. If the replicated computations are computed by different PEs at different times it is a space-time-shift scheme.

A systolic array with pipeline period $\alpha = 3$ can perform an original algorithm and redundant algorithms derived from the original one, concurrently. These redundant utations can be performed by the idle PEs at idle clock cycles. In the text that follows we will briefly describe the idea employed in [5] to design fault-tolerant hexagonal SA matrix multiplication.

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 [5]. The main idea in [5] was to achieve fault-tolerance through repeated computation of the same problem instance. The repeated computation is performed by employing space and/or time redundancy. 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

$$a(i-\tfrac{2r}{3}, j+\tfrac{r}{3}, k+\tfrac{r}{3}) := a(i-\tfrac{2r}{3}, j+\tfrac{r}{3}-1, k+\tfrac{r}{3});$$

$$b(i-\tfrac{2r}{3}, j+\tfrac{r}{3}, k+\tfrac{r}{3}) := b(i-\tfrac{2r}{3}-1, j+\tfrac{r}{3}, k+\tfrac{r}{3});$$

$$c(i-\tfrac{2r}{3}, j+\tfrac{r}{3}, k+\tfrac{r}{3}) := b(i-\tfrac{2r}{3}, j+\tfrac{r}{3}, k+\tfrac{r}{3}-1) +$$
$$c(i-\tfrac{2r}{3}, j+\tfrac{r}{3}, k+\tfrac{r}{3}) * b(i-\tfrac{2r}{3}, j+\tfrac{r}{3}, k+\tfrac{r}{3});$$

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 $a(i-\tfrac{2r}{3}, \tfrac{r}{3}, k+\tfrac{r}{3}) \equiv a_{ik}$, $b(-\tfrac{2r}{3}, j+\tfrac{r}{3}, k+\tfrac{r}{3}) \equiv b_{kj}$, and $c(i-\tfrac{2r}{3}, j+\tfrac{r}{3}, \tfrac{r}{3}) \equiv 0$. The output values of Algorithms_1 are $c(i-\tfrac{2r}{3}, j+\tfrac{r}{3}, N_3+\tfrac{r}{3}) \equiv c_{ij}$.

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

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

where data are used or computed, and a dependency matrix

$$D = [\vec{e}_b^3 \vec{e}_a^3 \vec{e}_c^3] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (2.1)$$

Using valid transformation matrix [5]

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

the computational structure (D, Pint (r)) is mapped into fault-tolerant SA, i.e.

$$T: (D, \text{Pint}(r)) \mapsto (\Delta, \overline{P}_{int}(r)), \quad (2,3)$$

where Pint(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, Pint(r)) into ($\Delta, \overline{P}$ int(r)), can be generated. For T defined by (2.2), the (x, y) positions of PEs in the SA are given by

$$\begin{bmatrix} x \\ y \end{bmatrix} = S \cdot \vec{p} = \begin{bmatrix} -1 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix} \cdot \begin{bmatrix} i - \frac{2r}{3} \\ j + \frac{r}{3} \\ k + \frac{r}{3} \end{bmatrix} = \begin{bmatrix} j - i + r \\ k - j \end{bmatrix}, \quad (\vec{p} \in P_{int}(r)) \quad (2,4)$$

while time schedule, t, is determined according to

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

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

$$\Delta_s = [\vec{e}_b^3 \vec{e}_a^3 \vec{e}_c^3] = \begin{bmatrix} -1 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix} \cdot D = \begin{bmatrix} -1 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix}. \quad (2,6)$$

The array obtained according to (2.4), (2.5) and (2.6) has the following features:

- number of PEs: np = N1 (N2 -1) + N2 (N3 + 1) + N3 (N1 + 1) -1,
- computation time: $t_c = N1 + N_2 + N_3 - 2$,
- AT factor: $AT = n_p \times t_c$ = [Nl(N2-1)+N2(N3+1)+N3(N1+1)-1] [N1+N2+N3-2].

For the case N1= N2 = N3 = N the following is obtained [4,5]:

$$n_p = 3N^2 + N - 1, t_c = 3N - 2,$$

$$AT = (3N^2 + N - 1)(3N - 2) = O(9N^3), \quad (2,7)$$

The corresponding SA that implements fault-tolerant matrix multiplication for $N_1 = N2 = N_3 = 3$ is shown in Fig.1. This array is referred to as optimal fault-tolerant SA with respect to AT factor in [5].

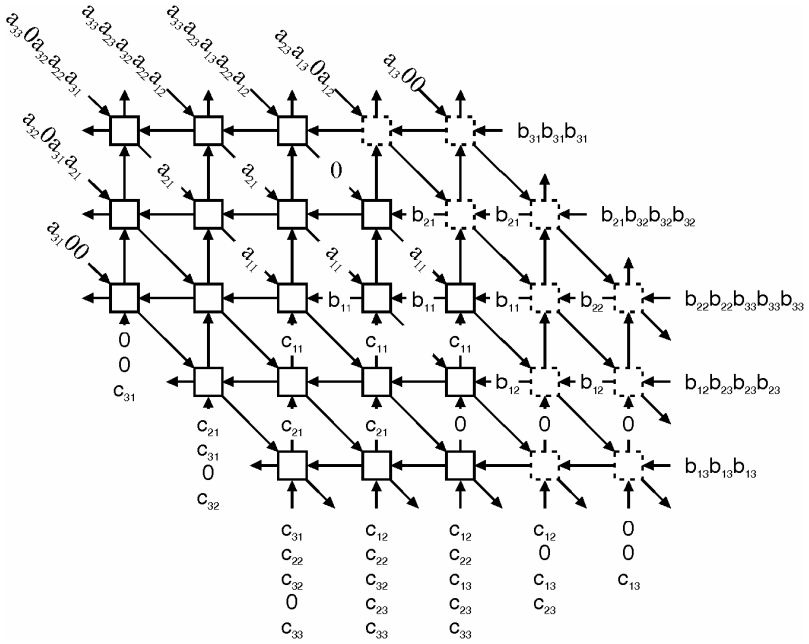

Figure1: Data flow in the SA during fault-tolerant matrix multiplication for N1 = N2 = N3 = 3, obtained in [5]

3. Our approach to mapping matrix multiplication algorithm onto fault-tolerant SA

This section describes a modification of the method given in [5] to synthesize a hexagonal SA which implements fault-tolerant matrix multiplication with minimal number of PEs with respect to the problem size. Although this array has slightly increased computation time compared to the one obtained in [5], its AT measure is almost two times less than that of the array given in [5]. Therefore we refer to this array as optimal fault-tolerant SA with respect to the number of PEs. To achieve this, we apply some linear mappings that accommodate the inner computation space Pint(r) to the projection direction $\vec{\mu} = [1\ 1\ 1]^T$ prior to the mapping defined by (2.3). The accommodation of Pint (r) can be performed both on index variable i and j. The choice depends on the relation between N1 and N2

(i.e. N1 > N2 or N1 < N2). When N1 = N2 both accommodations are equally profitable.

We will first consider the case when $N1 > N_2$, 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 N3 do

for j := 1 to N2 do

for i := 1 to N1 do

$$a(i+\tfrac{r}{3}, j-r, k) := a(i+\tfrac{r}{3}, j-r-1, k);$$

$$b(i+\tfrac{r}{3}, j-r, k) := b(i+\tfrac{r}{3}-1, j-r, k);$$

$$c(i+\tfrac{r}{3}, j-r, k) := c(i+\tfrac{r}{3}, j-r, k-1) + a(i+\tfrac{r}{3}, j-r, k) * b(i+\tfrac{r}{3}, j-r, k);$$

endfor {i, j, k, r};

With the following initial values $a(i+\tfrac{r}{3}, -r, k) \equiv a_{ik}$, $b(\tfrac{r}{3}, j-r, k) \equiv b_{kj}$, and $c(i+\tfrac{r}{3}, j-r, 0) \equiv 0$. The output values of Algorithms_2 are $c(i+\tfrac{r}{3}, j-r, N_3) \equiv c_{ij}$. Note that computations in Algorithms_2 satisfy all the conditions given in [5] to perform fault-tolerant matrix multiplication. The inner computation spaces of Algorithms_2 are given by

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

The data dependency matrix is the same as for Algorithms_1, i.e. as one given by (2.1). The accommodation of Pint (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 [8-10]) according to

$$H : P_{int}(r) \mapsto P^*_{int}(r),$$

i.e.

$$\vec{p}^{\,*} = F \cdot \vec{p} + G \qquad (3.1)$$

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

$$F = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}, \text{ and } G = \begin{bmatrix} 0 \\ -1 \\ -1 \end{bmatrix}. \qquad (3.2)$$

According to (3.1) the elements $\vec{p}^{*} = [u, v, w]$ of index space $P^{*}_{int}(r)$ are obtained as

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = F \cdot \begin{bmatrix} i + \frac{r}{3} \\ j - r \\ k \end{bmatrix} + G = \begin{bmatrix} i + \frac{r}{3} \\ i + j - \frac{2r}{3} - 1 \\ i + k + \frac{r}{3} - 1 \end{bmatrix}, \tag{3.3}$$

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

$$T : (D, P^{*}_{int}(r)) \mapsto (\Delta, \overline{P}_{int}(r)), \tag{3.4}$$

where the transformation T is given by

$$T = \begin{bmatrix} \vec{\Pi} \\ S \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix} \tag{3.5}$$

Note that transformation matrix T is not the same as one given by (2.2). Namely, under the conditions defined in [9], the transformation T given by (2.2) 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

$$\begin{bmatrix} x \\ y \end{bmatrix} = S \cdot \vec{p}^{*} = \begin{bmatrix} -1 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} i + \frac{r}{3} \\ i + j - \frac{2r}{3} - 1 \\ i + k + \frac{r}{3} - 1 \end{bmatrix} = \begin{bmatrix} j - r - 1 \\ k - 1 \end{bmatrix},$$

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

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

Note that for input data items we assume the following periodicity

$$a(i + \tfrac{r}{3} + N_1, j - \tfrac{2r}{3}, k + \tfrac{r}{3}) \equiv a(i + \tfrac{r}{3}, j - \tfrac{2r}{3}, k + \tfrac{r}{3} + N_3) \equiv a_{ik}$$
$$b(i + \tfrac{r}{3}, j - \tfrac{2r}{3} + N_2, k + \tfrac{r}{3}) \equiv b(i + \tfrac{r}{3}, j - \tfrac{2r}{3}, k + \tfrac{r}{3} + N_3) \equiv b_{ik} .$$

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

$$a(i + \tfrac{r}{3}, 0, i + k + \tfrac{r}{3} - 1) \mapsto \begin{bmatrix} x \\ y \end{bmatrix}_a = \begin{bmatrix} 3 - 3i - k - r \\ k - 2 \end{bmatrix}$$

$$b(0, i+j-1-\tfrac{2r}{3}, i+k+\tfrac{r}{3}-1) \mapsto \begin{bmatrix} x \\ y \end{bmatrix}_a = \begin{bmatrix} 3i+2j+k-r-5 \\ 3i+j+2k-5 \end{bmatrix},$$

$$c(i+\tfrac{2r}{3}, i+j-\tfrac{2r}{3}-1, 0) \mapsto \begin{bmatrix} x \\ y \end{bmatrix}_c = \begin{bmatrix} j-r-1 \\ 3-3i-j \end{bmatrix}$$

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

$$n_p = N_3 (N_2 + 2),$$
$$t_c = 3N_1 + N_2 + N_3 - 4, \qquad (3.6)$$
$$AT = N_3 (N_2 + 2)(3N_1 + N_2 + N_3 - 4)$$

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

$n_p = N(N+2)$, $t_c = 5N - 4$, $AT = N(N+2)(5N-4) = O(5N^3)$. (3.7)

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

$a(i-r, j+\tfrac{r}{3}, k) := a(i-r, j+\tfrac{r}{3}-1, k);$

$b(i-r, j+\tfrac{r}{3}, k) := b(i-r-1, j+\tfrac{r}{3}, k);$

$c(i-r, j+\tfrac{r}{3}, k) := c(i-r, j+\tfrac{r}{3}, k-1) + a(i-r, j+\tfrac{r}{3}, k) * b(i-r, j+\tfrac{r}{3}, k);$

endfor{i, j, k, r};

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

The inner computation spaces of Algorithms_3 are given by

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

Data dependency matrix is the same as one given by (2.1) . 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 {3.1}, with H = (F, G) being defined as [8-10]:

$$F=\begin{bmatrix}1 & 1 & 0\\ 0 & 1 & 0\\ 0 & 1 & 1\end{bmatrix},\ \text{and}\ \ G=\begin{bmatrix}-1\\ 0\\ -1\end{bmatrix}, \tag{3.8}$$

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 (3.8), the transformation T given by (3.5) is no more appropriate one (see for example [9]). According to the criteria established in [9], for T we can choose

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

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

$$\begin{bmatrix}x\\ y\end{bmatrix}=S\cdot\vec{p}^{\,*}=\begin{bmatrix}1 & -1 & 0\\ 0 & 1 & -1\end{bmatrix}\cdot\begin{bmatrix}i+j-\frac{2r}{3}-1\\ j+\frac{r}{3}\\ j+k+\frac{r}{3}-1\end{bmatrix}=\begin{bmatrix}i-r-1\\ 1-k\end{bmatrix},$$

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

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

Note that for input data items we assume the following periodicity

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

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

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

$$a(i+j-\tfrac{2r}{3}-1,0,j+k+\tfrac{r}{3}-1)\mapsto\begin{bmatrix}x\\ y\end{bmatrix}_a=\begin{bmatrix}2i+3j+k-r-5\\ 5-3j-i-2k\end{bmatrix}$$

$$b(0,j+\tfrac{r}{3},\ j+k+\tfrac{r}{3}-1)\mapsto\begin{bmatrix}x\\ y\end{bmatrix}_b=\begin{bmatrix}3-3j-k-r\\ 1-k\end{bmatrix},$$

$$c\ (i+j-\tfrac{2r}{3}-1,\ j+k+\tfrac{r}{3},0) \mapsto \begin{bmatrix} x \\ y \end{bmatrix}_c = \begin{bmatrix} i-r-1 \\ 3j+i-3 \end{bmatrix}$$

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

$$n_p = N_3(N_1+2),$$
$$t_c = N_1 + 3N_2 + N_3 - 4, \qquad (3.9)$$
$$AT = N_3(N_1+2)(N_1+3N_2+N_3-4).$$

For $N_1 = N_2 = N_3 = N$ we obtain the same as in (3.7).

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

$n_p = N3\ (\min\{N_1, N_2\} + 2)$,

$t_c = (3 \max\{N_1, N2\} + \min\{N_1, N_2\} + N_3 - 4)$

$AT = N_3\ (\min\{N_1, N2\} + 2)\ (3 \max\{N_1, N2\} + \min\{N_1, N2\} + N3 - 4)$.

Obviously, for $N_1 = N_2 = N_3 = N$, features of the obtained array are determined by (3.7). 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 [5], 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 Fig.2. A detail concerning the voting mechanism is sketched in Fig.3. Fig. 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 $\left[\frac{N+2}{3}\right]*3$ multiplexers and $\left[\frac{N+2}{3}\right]$ 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.

A better error coverage can be achieved if voting is performed after each computational step. In this case additional hardware have to be inserted between each row of processing elements. Fig. 4. shows the structure of the voting hardware inserted between each two rows of the PEs. Now, there are two levels of multiplexers: $\left[\frac{N+2}{3}\right]*3$ at the output of PEs, and N+2 at the input of PEs.

plexers: $\left[\frac{N+2}{3}\right]*3$ at the output of PEs, and N+2 at the input of PEs. Control signals for all multiplexers in one row are unique and could be pipelined along with the elements of matrix C through the array. Again, the inputs of multiplexers are selected in a "round robin" way, starting from input 0. Each voter takes three results to vote and broadcast the result to three multiplexers. By the proposed scheme multiple faults occurred on the same resulting element can be tolerated if they appear at different clock cycles.

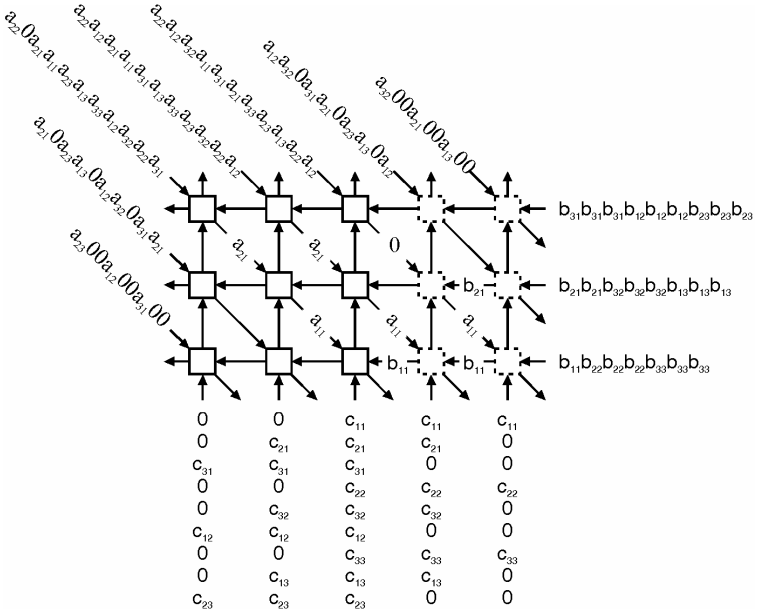

Figure 2: Data flow in the SA synthesized by the described procedure during fault-tolerant matrix multiplication for $N_1 = N_2 = N_3 = 3$

4. 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. Its AT factor is almost two times less than that of the array given in [5]. 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. We have described two hardware solutions for the voting process. One when voting is performed at the end of the computation, and the other when it is performed after each computational step. In both cases fault detection and location are not neces-

sary for fault-tolerance, errors are masked concurrently with normal operation of the systolic array.

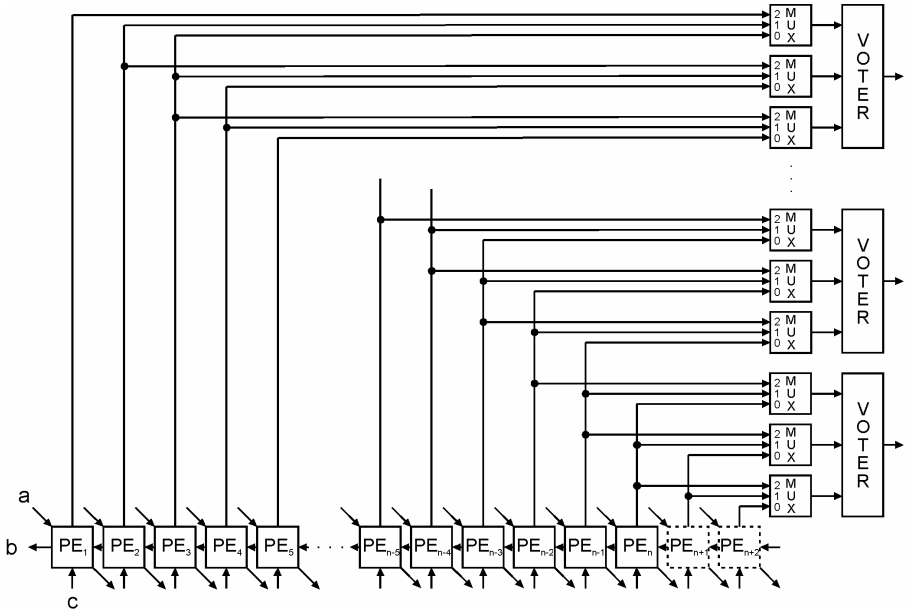

Figure 3: A detail of the voting process

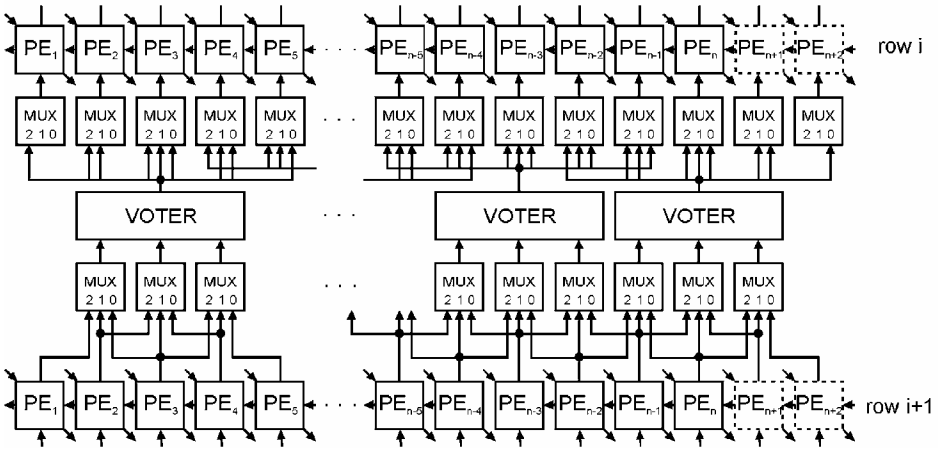

Figure 4: A detail of the voting process when performed between each stage of processing elements

5. 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. Stojcev, E. I. Milovanović, I. Ž. Milovanovic, 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, Inter. Conf. Parallel and Distributed Systems, Taiwan, 1994, 348-353.
5. 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
6. I. Z. Milentijević, I. Ž. Milovanovic, 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.
7. E. I. Milovanović, I. Z. Milentijević, I. Ž. Milovanović, Designing of processor-time optimal systolic array for matrix multiplication, Comput. Artificial Intelligence, Vo1.16, 1 (1997), 1-11.
8. E. I. Milovanović, G. V. Milovanović, I. Ž. Milovanović, D. Milosavljevic, Designing hexagonal systolic array by composite mappings, Facta Univesitatis (Nis), Ser. Math. Inform., Vol. 12 (1997), 283-296.
9. T. I. Tokić, I. 2. 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.
10. C. N. Zhang, Systematic design of systolic arrays for computing multiple problem instances, Microelectronics Journal, 23 (1992), 543-553.
11. J. -J. Wang, C.W. Jen, Redundancy design for a fault tolerant systolic array, IEE Proceedings, Vol. 137. Pt. E, 3 (1990), 218-226.