A Real-Time Reconfiguration Algorithm for Fault-Tolerant VLSI and WSI Arrays

Hussain Al-Asaad and Mankuan Vai

Department of Electrical and Computer Engineering
Northeastern University
Boston, MA 02115

Abstract

Reliability is an important issue in the real-time operations of VLSI array processors. A new algorithm for the real-time reconfiguration of VLSI and WSI arrays is presented. This algorithm is characterized by its simplicity and locality. The control of this reconfiguration scheme is implemented in hardware for a real time execution. It supports multiple faults including transient/intermittent faults with a zero degradation time. Simulation results show that a good spare utilization rate is achieved with a computational complexity that is independent of the array size.

1: Introduction

Due to the continuous increase in the density of integration and the size of array processors, the probability of having faults in a VLSI array becomes higher. If a processor array is not fault-tolerant, the failure of a processing element (PE) may lead to the fatal failure of the entire array. Fault-tolerance is thus an important feature for array processors. Fault tolerance is typically achieved by adding redundant PE's to the array and, when a fault occurs, using reconfiguration techniques to reconfigure the array. In recent years, many reconfiguration schemes have been proposed [e.g., 1-6], however, only a few of them have considered the case of real-time reconfiguration. In this paper we present a very fast algorithm for real-time reconfiguration.

According to [7], there are four factors that characterize a reconfiguration algorithm: simplicity, efficiency, area, and locality. In real-time applications, reconfiguration has to be carried out as fast as possible. Hence, the most important factors in real-time reconfiguration are simplicity and locality. A simple algorithm can be executed in a very short time and local communication implies short propagation delays. These requirements can only be achieved by a local spare distribution scheme.

Real time reconfiguration has been considered in [4] and [6], both of which use a global redundancy scheme with one spare row and one spare column. Transient/intermittent faults are not supported in [4]. Also, multiple faults are not supported efficiently. The algorithm described in [4] is complicated when it is compared to, for example, the algorithm presented in this paper. On the other hand, multiple faults are not supported in [6] and its support of transient/intermittent faults requires a roll-back time equal to the reconfiguration time. A common shortcoming of these schemes is their inefficiency and the global distribution of spares leads to a speed degradation after a reconfiguration. The algorithm presented in this paper satisfies all real time reconfiguration requirements.

0-8186-2837-5/92 $03.00 © 1992 IEEE

2: Fault model

The following fault model is adapted in this paper. First, only random faults are considered. This assumption is acceptable in real time reconfiguration because clustered faults are typically caused by the fabrication process and are treated by off-line algorithms. Second, faults are considered to associate only with PE's. Interconnections are assumed to be fault free due to the random fault distribution assumption. This is a reasonable assumption because the area occupied by interconnections is much smaller than that of the processing elements. Third, reconfiguration controlling and switching elements are assumed to be fault free. Fourth, the self-testing hardware is also assumed to be fault free. While only the reconfiguration of rectangular arrays is discussed, the scheme is readily modifiable to adapt other topologies. The objective is to map a logical array of a given size onto a physical array.

3: Array structure

The array structure considered in this paper is an array provided with interstitial redundancy, which is similar to the one discussed in [5]. Spares are inserted in interstitial sites within the array. An example of a 3 × 4 array is shown in Fig. 1. The redundancy ratio of this reconfiguration scheme is approximately 100%, which is comparable to that of a duplicate system.

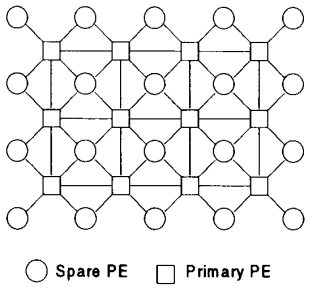

Fig. 1 A 3 × 4 array provided with interstitial redundancy.

Each PE consists of a self-testing block, a reconfiguration block, and a functional block. The self-testing block of a PE continuously tests its associated functional block and informs the reconfiguration block immediately after a fault is detected. The function of the reconfiguration block is to replace the faulty PE with a spare. Both spare and primary PE's have identical self-testing blocks, but their reconfiguration blocks are different.

Figs. 2a and 2b show the communications between the self-testing block and the reconfiguration block of a primary PE and a spare PE, respectively. A signal FP goes from the self-testing block of a primary PE to its reconfiguration block and is used to indicate the existence of a fault. A signal serving a similar function, called FS, is provided in a spare PE. The set of one primary PE and its four spare PE neighbors or the set of one spare and its four primary PE neighbors are referred to as a group in the following discussion. Examples of these groups are shown in Fig. 3.

The reconfiguration signals provided between a primary PE and a spare PE are shown in Fig. 4 and defined as:

SREQ (Service Request):

This signal goes from a primary PE to a spare PE. Each primary PE has four SREQ signals, each of which goes to a spare within its group.

SACK (Service Acknowledgment):

This signal goes from a spare PE to a primary PE. Each spare has four SACK signals, each of which goes to a primary PE within its group.

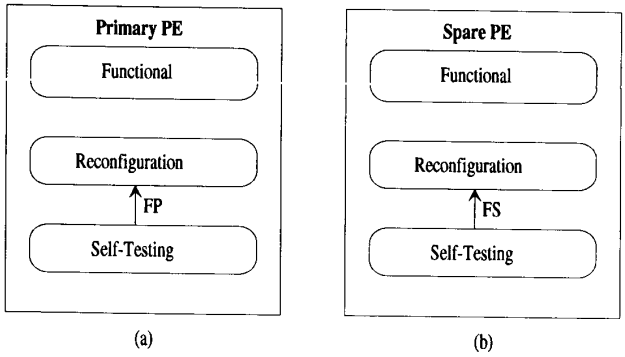

Fig. 2 The block diagrams of (a) a primary PE and (b) a spare PE.

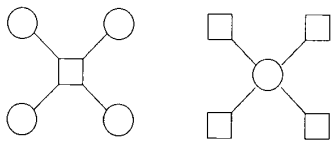

Fig. 3 Groups of PE's.

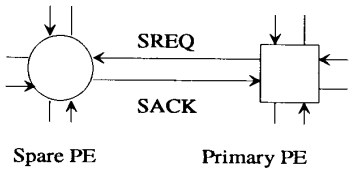

Fig. 4 Signals between a spare PE and a primary PE.

A spare PE uses its **SACK** line to indicate if it is available. An activated **SACK** line indicates that the spare is not available, which means that the spare is either faulty or being used to replace

a primary PE. The signal **SREQ** is used by a primary PE to request the service of a spare. A primary PE has to check the status of a spare by testing the corresponding **SACK** line before its service is requested.

4: Real-time reconfiguration

Now the reconfiguration algorithm itself is described. If a spare detects an internal fault by its self-testing block, a procedure **SFAULT** is executed. This procedure simply activates the **SACK** lines of a spare PE to indicate to its primary PE neighbors that it is faulty.

A **PFAULT** procedure is executed in a primary PE after a fault is detected. The **PFAULT** procedure checks the location of the fault. If the fault is within the primary PE (i.e., one detected by its self-testing block), another procedure **SWITCH** is executed to select one of the available spares according to certain criteria (to be described later) and activate the **SREQ** line connected to the selected spare. On the other hand, when the **SACK** line of a spare PE is activated to indicate that it is unavailable, the following operations are carried out. If the **SREQ** line connecting the primary PE to this unavailable spare is negated, indicating that this spare is not a substitute of the primary PE, then no action is needed. Otherwise, an activated **SREQ** line connecting the primary PE to this spare PE shows that the spare PE previously selected to replace the faulty primary PE has also become faulty. The procedure will check to decide if the primary PE has recovered. If the primary PE has recovered from a transient/intermittent fault, it can now resume its operations. If the test result is still negative, the procedure **SWITCH** is activated to select another available spare.

The procedure **SWITCH** is responsible for finding a spare PE to replace a faulty primary PE. This spare PE must be healthy and belongs to the four neighbor spare PE's around the primary PE. When more than one healthy spare is available to replace a failed primary PE, a method is needed to systematically select one out of them. An inadequate selection will affect the result of later selections. The goal of this selection is to achieve a maximum spare utilization. The following methods of selection have been evaluated.

a) RANDOM selection (**RAND**):

The advantage of this method is its simplicity. In this method a spare PE is randomly selected from the available spare PE's.

b) ROTATING selection (**ROTA/ROTB**):

In this method the first available spare PE, found by going in the clockwise direction around the primary PE (see Fig. 5), is selected. Two schemes, characterized by their starting points, are possible:

i) **ROTA:** Always start from the spare PE located at the upper left corner of the primary PE. This starting point is highlighted in Fig. 6.

ii) **ROTB**: Evenly divide the array into four regions as shown in Fig. 7. The starting point in each region is shown in Fig. 8.

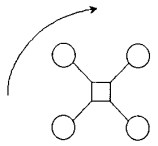

Fig. 5 The direction of spare PE selection.

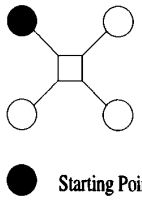

Fig. 6 The starting point of spare PE selection.

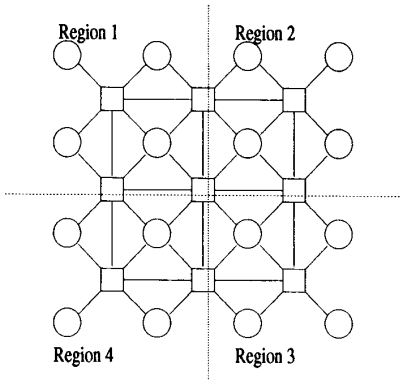

Fig. 7 The division of an array into four regions.

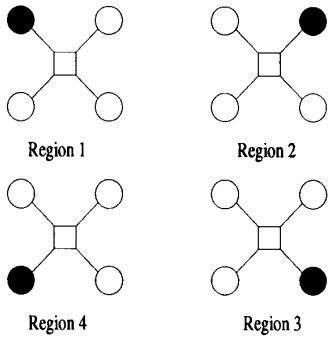

Fig. 8 The starting points of spare PE selection in different regions.

c) PREDICTIVE selection (**PRED**):

This scheme is relatively complicated but has the benefit of obtaining a higher rate of spare utilization. In this method each spare PE maintains a number N which is the number of healthy primary PE's within its group. Once the procedure **SWITCH** is executed, the primary PE will collect the value of N from each of the healthy spares within its group. The spare PE with the highest N value is selected. The spares located at the boundary of the physical array have less than four primary PE's and will treat the non-existing primary PE's as always healthy. In other words, all the spares initially have a value of N = 4. A RANDOM selection or ROTATING selection could be used to pick a spare from those having equal N values.

Figs. 9 and 10 show the **ROTA** reconfiguration control circuits for the spare PE and the primary PE, respectively. The reconfiguration part in a spare is designed as a pure combinational circuit, while that of the primary PE is designed as a sequential circuit. These control circuits can be easily adapted to other selection schemes by rearranging the control signal lines.

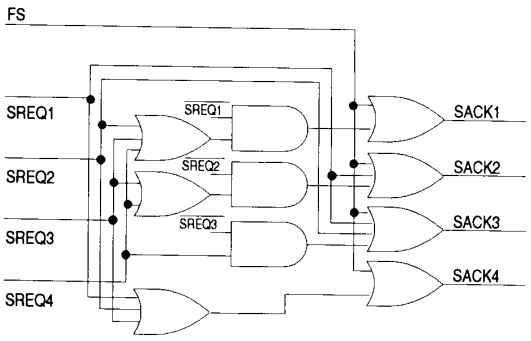

Fig. 9 The ROTA control circuit of the reconfiguration block in a spare PE.

An important characteristic of this algorithm is that the recovery of transient/intermittent faults is done with zero degradation time, i.e., without interrupting the operation of the array. The self-testing block of a faulty PE (primary or spare) continues to check the condition of its functional block. If a spare PE recovers from a transient/intermittent fault, it will negate its **SACK** lines to indicate to the primary PE's in its group that it could be used in the future. The negation of a **SACK** line does not affect the operation of the corresponding primary PE. On the other hand, if a primary PE becomes healthy again after hit by a transient/intermittent fault, the previously chosen spare will continue to act as a replacement as long as it is healthy. In other words, the recovered primary PE will only be used again if the selected spare PE fails.

This real-time reconfiguration algorithm is simulated and evaluated. In the simulation permanent faults are randomly injected in the array. Simulation results are shown in Table 1. In this table, the survival probabilities of a 10 × 10 array using the above described reconfiguration schemes are given as functions of the number of failed PE's. The first column shows the number of failed PE's normalized by the total number of spares. Columns 2-5 show the probabilities of survival for this algorithm using the four selection methods discussed above, respectively. In order to compare the performance of this algorithm with an existing real-time reconfiguration

algorithm, the probabilities of survival using the global algorithm presented in [4] are also included. This simulation demonstrates that a comparatively good spare utilization is achieved in real-time by a simple local algorithm. Another indication of improvement is that the minimum number of faults that cause a fatal failure has been increased from three in [4] to five in the algorithm presented in this paper.

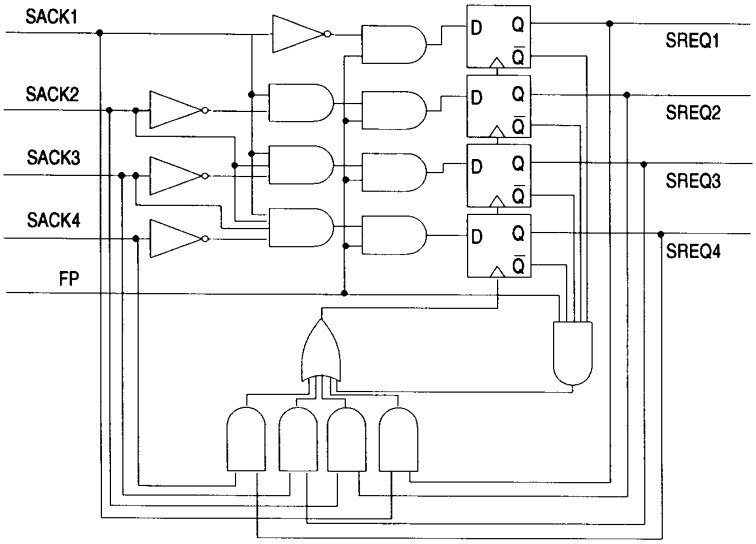

Fig. 10 The ROTA control circuit of the reconfiguration block in a primary PE.

No. of Failed PE's (Normalized by the No. of spares)	RAND	ROTA	ROTB	PRED	Algorithm of [4]
0.1	1.00	1.00	1.00	1.00	1.00
0.2	0.99	1.00	0.99	1.00	0.99
0.3	0.95	0.95	0.94	0.97	0.93
0.4	0.82	0.80	0.79	0.85	0.81
0.5	0.48	0.48	0.47	0.58	0.53
0.6	0.16	0.15	0.15	0.22	0.32
0.7	0.02	0.01	0.01	0.04	0.13
0.8	0.00	0.00	0.00	0.00	0.03

Table 1 Probability of survival in a 10 × 10 processor array.

5: Summary

In summary, a very fast real-time reconfiguration algorithm for fault-tolerant VLSI and WSI arrays is developed. This algorithm is featured by its locality and simplicity, and can be fully implemented in hardware for very fast execution. It supports multiple faults including transient/intermittent faults with a zero degradation time. A good spare utilization is achieved with a computational complexity of O(1).

6: References:

[1] Distante, F., Sami, M.G., Stefanelli, R., "A general index mapping technique for array reconfiguration," *Proc. ISCAS-88*,1988, pp. 559-563.

[2] Leonardo, J., Lombardi, F., Sciuto, D., "Orthogonal mapping: A reconfiguration strategy for fault tolerant VLSI/WSI 2-D arrays," *Proc. Int'l Workshop on Defect and Fault Tolerance in VLSI Systems*, 1988, pp. 309-318.

[3] Lombardi, F., Sami, M.G., Stefanelli, R., "Reconfiguration of VLSI arrays by covering," *Transactions on Computer-Aided Design*, Vol. 8, No. 9, September 1989, pp. 952-965.

[4] Sciuto, D., Lombardi, F., "Real time reconfiguration of two dimensional VLSI arrays," *Proc. Euromicro Workshop Real Time 1989*, N.J., USA, pp. 217-225.

[5] Singh, A., "Interstitial redundancy: An area efficient fault tolerance scheme for large area VLSI processor arrays," *Transactions on Computers*, Vol. 37, No. 11, Nov. 1988.

[6] Kung, S.Y., Chang, C.W., Jen, C.W., " Real-time reconfiguration for fault-tolerant VLSI array processors," *Real Time Systems Symposium*, 1986, pp. 46-54.

[7] Chean, M., Fortes, J., "A taxonomy of reconfiguration techniques for fault tolerant processor arrays," *Computer*, Vol. 23, No. 1, January 1990, pp. 55-69.