VLSI DESIGN
1998, Vol. 7, No. 2, pp. 143–150
Reprints available directly from the publisher
Photocopying permitted by license only

© 1998 OPA (Overseas Publishers Association)
Amsterdam B.V. Published under license
under the Gordon and Breach Science
Publishers imprint.
Printed in India.

Characterization of Catastrophic Faults in Reconfigurable Systolic Arrays

VINCENZO ACCIARO and AMIYA NAYAK*

School of Computer Science, Carleton University, Ottawa, Canada, K1S 5B6

(Received 26 February 1996)

A common technique widely used to achieve fault tolerance in systolic arrays consists in incorporating in the array additional processing elements (PEs) and extra bypass links. Given a sufficient number of PEs and a large enough set of bypass links, it might seem that the array can easily tolerate a large number of faults provided they do not occur in consecutive locations. It is not always the case as shown in this paper. In fact, certain fault patterns exist and may occur which would prevent any kind of restructuring of the aray, thus making the structure unusable. For a given set of bypass links from each PE in the array, it is possible to identify such fault patterns which will prevent any reconfiguration. In this paper, we identify the class of fault patterns that are catastrophic for linear systolic arrays, examine their characteristics, and describe a method for constructing such fault patterns.

Keywords: Systolic arrays, catastrophic fault patterns, fault tolerance

1. INTRODUCTION

Systolic arrays are effective means to achieve significant speedups in execution time for computation-bound problems by exploiting parallel processing and pipelining [5]. Proper fault tolerance mechanisms must be in place to cope with the chip-related failures in such arrays. The basic sources of chip-related failures for VLSI are either process related or design related [8]. The process related defects are those which happen during the integrated circuit processing while the design related defects are those which show up during the design phase of the chip. A popular approach for achieving fault tolerance in VLSI-based systolic arrays is by incorporating redundancy. With modern technology it is now possible to incorporate a large degree of redundant PEs and additional circuitry into a single chip. Furthermore, the regular structure of systolic arrays makes it even easier to add extra PEs and links. The redundant PEs can be used to replace any faulty PEs which occur in the active part of the array. However, more number of PEs in a chip would require an extra overhead of interconnections and switching circuitry which would mean a

*Corresponding author.

high likelihood of failure. The redundant links can be used to bypass one or more faulty PEs. "Long" interconnection would mean that it would be possible to bypass larger blocks of consecutive faulty PEs. At first, it might seem to be theoretically possible to achieve a very high degree of fault tolerance, simply by providing a sufficiently large number of spare PEs with a large number of long connections. Unfortunately, in practice, this is not feasible because of different layout constraints and signal propagation delay. In fact, signals from one PE to another in a path will encounter additional propagation delay as the connections between PEs are made longer than before; this might, in turn, create synchronization problems. Minimizing the length of the longest connection is important because signal propagation delay can be the limiting factor in the performance of the system [3, 8].

The nearest neighbourhood porperty must be maintained at all times in the event of PE or link failures for the structure to be useful. This requires reconfiguration which is essentially the process of restructuring the array in which faulty elements are mapped to the spares while preserving high degree of regularity and locality of reference. The effectiveness of a fault tolerance scheme can be measured not just by the amount of redundancy built into the array, but also the ability to restructure the array in presence of faults and carry on operation. A large number of redundant PEs would mean nothing if they cannot be used successfully to replace faulty ones. Reconfiguration may require bypass links to span several rows and/or columns and can fail due to the lack of sufficient spares and suitable bypass links to replace faulty ones and to go around the faulty area. In past decade, many researchers have proposed several fault tolerance techniques, interconnection strategies, and reconfiguration schemes for VLSI-based systolic arrays and various other parallel architectures [1–4, 6, 7, 11].

VLSI-based systolic arrays are normally designed with a fixed number of spare PEs and redundant links of fixed lengths. Restriction on propagation delay limits the length of bypass links. Given a sufficient number of PEs and a large enough set of bypass links, it might seem that the array can easily tolerate a large number of faults provided they do not occur in consecutive locations (a block fault). Even with sufficiently long bypass links, faults at strategic locations can have catastrophic effect on the entire structure and cannot be overcome by any amount of clever design. For a given set of bypass links from each PE in the array, it is possible to identify such fault patterns which will prevent any reconfiguration. In this paper, we identify the class of fault patterns that are catastrophic for one-dimensional systolic arrays, examine their characteristics and describe a method for constructing such fault patterns.

The paper is organized as follows. Section 2 gives basic concepts, error model, and relevant assumptions that provide basis for further analysis. In Section 3, we characterize catastrophic fault patterns in terms of the properties of a class of integer sets. Finally, in Section 4, we describe a method for generating all catastrophic fault patterns using a graph theoretical approach.

2. PRELIMINARIES

In this paper, we will focus on one-dimensional (or linear) arrays. The basic component of a systolic array is the processing element (PE) as shown in Figure 1. Each PE in this array consists of a multiplier, an adder, some control registers, and some logic and fault detection circuits. The peripheral PEs are responsible for I/O functions. The links can be either unidirectional (i.e., in which case information flows only in one direction) or bidirectional (the links can be used in either direction). There are two kind of links in redundant arrays. They can be either regular or

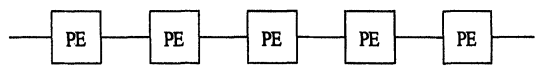


FIGURE 1 One dimensional array of processors.

bypass. Regular links exist between neighbouring PEs while the bypass links are assumed to exist between non-neighbours. The bypass links are used strictly for reconfiguration purposes when a fault is detected. For all other purpose, the bypass links are considered to be the redundant links.

Figure 1 shows a simple case of one dimensional structure. Bypass links are not shown for simplicity. A fault in a PE can be best described as "its inability to perform correct operation on correct inputs", possibly due to some elementary circuit or gate-level faults. The failures could be either permanent or transient. In this paper, we will consider only with process related permanent random and block failures at PE level. The PEs not active in the absence of faults are called spares. They have no specific designation that distinguishes them from the active PEs. The location of spares is irrelevant for our discussion.

DEFINITION 1 We say that a bypass link has length g if it connects two PEs which are of length g apart. In other words, they have $g-1$ PEs in between. The regular links are of length 1.

For a linear array with no link redundancy, a single PE fault in any location is sufficient to stop the flow of information from one side to the other. This fact holds irrespective of the PE redundancy. We can make similar observations in situations where there is some form of link redundancy. It is again obvious that a linear array with bypass links $\{g_1, g_2,\ldots,g_k\}$ cannot tolerate g_k PE faults if they occur in a block (or cluster). In order to tolerate a block fault of size g_k, we must have some bypass links of length at least g_k+1. This can be illustrated by the following simple example.

Example 1 Figure 2 shows a linear array in which, each PE is connected to its immediate neighbours and to the PEs at distance 2 and 4. The faulty PEs are shadowed. Clearly, there is no single link or a combination of links that would allow us to bypass the faulty area. In other words, the structure is logically separated for all practical purposes.

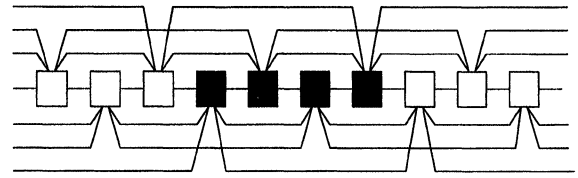


FIGURE 2 A block fault of size 4 for links {1, 2, 4}.

It is interesting to observe that the number of faults sufficient to partition the array is a function only of the length of the longest bypass connection, g_k, but not of any intermediate bypass connections. In practice, the probability of block faults of size g_k or higher is relatively small. A natural question is whether the bypass links $\{g_2,\ldots,g_k\}$ will be sufficient to cope with g_k faults if they do not occur in a block. Unfortunately, there exist certain fault distributions involving g_k faults, not all in a block, for which the structure will be unusable. The following example will demonstrate that.

Example 2 Once again consider the same linear array as in the previous example with the same number of faults, but a different fault distribution (see Fig. 3). In this case, it is not difficult to verify that any link from a non-faulty PE outside the faulty region either leads to a faulty PE directly or to a non-faulty PE within the faulty region which then eventually leads to a faulty PE.

Let $G=\{g_1,\ldots,g_k\}$ be the set of links (link configuration) with $g_i<g_{i+1}$ for the linear array. We assume $g_1=1$ to be the regular link connecting two adjacent neighbours. Let $F=\{f_1,\ldots,f_m\}$ be the ordered set of faulty PE's.

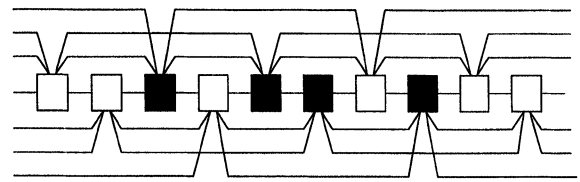


FIGURE 3 Fault pattern for $g_k=4$ faults.

DEFINITION 2 A fault pattern of size m is an assignment of m faults in the array. A faulty region is the smallest block containing all faulty PEs. An example of fault pattern of size 4 given in Figure 3.

DEFINITION 3 The width W_F of a fault pattern F if the number of PEs between and including the first and the last fault in F. That is, $W_F = f_m - f_1 + 1$.

DEFINITION 4 We say that a fault pattern is catastrophic with respect to a given link configuration when the peripheral PEs are no longer connected. It should be noted that not all fault patterns are catastrophic.

The width of a catastrophic fault pattern depends on two quantities: the link configuration (the number of links including the bypass links) G and the orientation (unidirectional or bidirectional links) of G. Obviously the lower bound on the width of a catastrophic fault pattern, consisting of g_k number of faults, is g_k. This is the case when all faults occur in a cluster. The upper bound on the window size has been established in [9, 10] and is given by:

$$W_F \leq \begin{cases} (g_k - 1)^2 + 1 & \text{for unidirectional links} \\ (\lceil g_k/2 \rceil - 1)g_k + \lfloor g_k/2 \rfloor + 1 & \text{for bidirectional links} \end{cases}$$

In our paper we consider only fault patterns which occur in a window of length $2g_k$. We assume that all the PEs of the left and all the PEs on the right of the window are same (non-faulty). Under this assumption we call such fault patterns *localized.*

3. PROPERTIES OF LOCALIZED FAULT PATTERNS

Assuming an infinite number of PEs, the natural ordering induced by the regular links defines a one to one correspondence between the PEs and the rational integers. Note that in this case a fault pattern is catastrophic iff it disconnects the array into two infinite components. The minimum number of faults required to construct a catastrophic fault pattern is a function of the length of the longest bypass link, not the number of bypass links as it might seem at the first glance. Furthermore, we show that certain symmetric property must hold within the window, for a fault pattern to be catastrophic with respect to a given set of links.

DEFINITION 5 Given G and F, define D to be the ordered of unreachable integers as follows: $i \in D$ iff $i \in F$, or $i+g \in D \; \forall \, g \in G$, or $i - g \in D \; \forall g \in G$.

DEFINITION 6 If $D = Z$ then F is said to be blocking for G, where Z is the set of all integers.

THEOREM 1 *F is a catastrophic fault pattern for the linear array if and only if F is blocking for G.*

Proof $D \neq Z$ iff there is an $l \notin D$ iff $l + g_{r_1} \notin D$ for some g_{r_1} iff $l + g_{r_1} + g_{r_2} \notin D$ for some g_{r_2} iff ... iff $b = l + g_{r_1} + g_{r_2} + \ldots + g_{r_j} > f_n$ for some g_{r_j}. But $l \notin D$ iff $l - g_{l_1} \notin D$ for some g_{l_1} iff $l - g_{l_1} - g_{l_2} \notin D$ for some g_{l_2} iff ... iff $a = l - g_{l_1} - g_{l_2} - \ldots - g_{l_k} < f_1$ for some g_{l_k}. The sequence of links $g_{l_k}, \ldots, g_{l_1}, g_{r_i} \ldots, g_{r_j}$ connect a to b: this is equivalent to assert that the graph is not disconnected into two infinite components. □

Since the bijection between Z and the PEs is defined by giving the PE corresponding to $0 \in Z$, we obtain the following:

PROPERTY 1 If F is blocking for G then for any integer c, $F + c$ is blocking for G.

This tells us that if a fault pattern is blocking (catastrophic) then the occurence of this pattern anywhere within the array will still be blocking. That is, the window can be translated along the integer axis.

From this property it follows that without loss of generality we can restrict ourselves to the case in which the g_k faults are integers in $Z[2g_k]$ where $Z[x] := \{1, \ldots, x\}$ for $x \in Z^+$.

Let $S = Z[2g_k] - F$ be the set of non-faulty or sane integers in the "window".

THEOREM 2 *F is blocking for G if and only if* $i+g \in F$, *for all* $i \in S \cap Z[g_k]$ *and* $g \in G$ *such that* $i+g > g_k$.

Proof (only if part) By contradiction assume that there exist $i \in Z[g_k]$ and $g \in G$ such that $i+g > g_k$ and both i and $i+g$ are in S. It is easy to see that the infinite sequence of integers..., $i-j \cdot g_k, \ldots, i, i+g, i+g+g_k, \ldots, i+g+j \cdot g_k, \ldots$ is not in D for $j=0,1,2,\ldots$ contradicting the fact that F is blocking for G.

(if part) It will be shown by induction that, if the conditions stated in the theorem hold, all integers in S are in D. Let i be the largest in S. By definition of D, the conditions of the theorem imply that $i \in D$; this forms the basis for the inductive argument. Let all the p largest integers in S be also in D; we will now show that also the $(p+1)^{th}$ largest integer in S is also in D. Let i be such an integer in S and let $g \in G$ be such that $i+g$ is not in F. If $i+g > g_k$ then by definition $i+g \in D$; if $i+g \leq g_k$, then $i+g \in D$ by inductive hypothesis. Thus, for every $g \in G$, $i+g \in D$. The condition that all integers in S to be in D clearly implies Z^+ is in D which in turn implies that all integers are in D. Thus F is blocking for G. □

The theorem can be explained as follows. Consider the non-faulty or sane elements in the first half of the window; i.e., in $Z[g_k]$. For the fault-pattern F to be blocking for G, every non-faulty element in the first half of the window must lead us directly to a faulty element in the second half and vice versa.

COROLLARY 1 *Let* $i \in Z[g_k]$. *Then* $i \in S$ *if and only if* $i+g_k \in F$.

Proof Let $g=g_k$ in the theorem above. □

This corollary is the symmetric property of the window and tells us that when an element in the first half of the window is non-faulty its corresponding element in the second half of the window will have a corresponding element which is faulty.

Example 3 Figures 4 and 5 shows two fault patterns which are contained in an interval of size $2g_k$ for link configuration $G=\{1, 3, 7, 10\}$. Both fault patterns are symmetric, that is if an element in the first half of the window (interval) is faulty then the corresponding element in the same location of the second half of the window is non-faulty.

FIGURE 4 A symmetric fault pattern for $G=\{1,3,7,10\}$.

FIGURE 5 A symmetric fault pattern not blocking for $G=\{1, 3, 7, 10\}$.

It should be noted that the symmetric property is a necessary condition but it is not a sufficient condition. That is, a symmetric fault pattern is not necessarily catastrophic. For example, the fault pattern given in Figure 5 is not blocking for G although it is symmetric.

The theorem and the corollary imply that a blocking set F is completely characterized by the set W of non-faulty integers in the first half of the window, i.e., by $S \cap Z[g_k]$.

DEFINITION 7 Given a subset W of $Z[g_k]$ let $F[W]:=(Z[g_k] - W) \cup (W+ g_k)$. W is called a solution for G if $F[W]$ is blocking for G. $F[W]$ is essentially the set of faulty elements in the entire window.

That is, to each solution corresponds a unique fault pattern and vice versa. Thus, by knowing all possible solutions, we can generate all possible fault patterns. Additional property given in Corollary 2 must hold in order for an integer set to be a solution.

COROLLARY 2 *W is a solution if and only if for all* $i \in W$ *and for all* $g \in G$ *such that* $i+g > g_k$, $i+g \bmod g_k \in W$.

Proof The corollary is a direct consequence of Theorem 2 and Corollary 1. □

This corollary illustrates the property that the elements in a solution are closed under modulus; this property will be used to verify if an integer set is a solution. Later, it will be shown that the set of solutions is closed under union.

Example 4 Consider the set of links $G=\{1, 5, 7, 9\}$ and two sets of integers $W_1 = (2,4,6,8)$ and $W_2=(1,2,5,7,9)$. Use of Corollary 2 yields that W_1 is a solution for G, but W_2 is not.

As a consequence of the corollary the presence of integer i in a solution W "forces" other integers to be in W i.e., the integers $i+g$ mod g_k whenever $i+g>g_k$.

DEFINITION 8 Let $W(i)$ be the set of all integers forced by i in W. It is not difficult to verify that $W(i)$ is a solution. Denote the *base set solution by* $\{W(i)|i= 1,2,\ldots,g_k\}$.

We now present an algorithm for the construction of $W(i)$. The algorithm is very simple and finds a solution forced by some integer element i.

Algorithm for the construction of $W(i)$

```
Input : G and i
Output : W(i)
Begin
  W ← ∅
  P ← i
  While nonempty P do
    W ← j ← P
    For all g ∈ G such that g + j > g_k > g do
    P ← g + j − g_k
  EndFor
EndWhile
End
```

Repeated executions of the algorithm with $i= 1,2,\ldots,g_k$ will construct the base set. The construction can actually be simplified by using the following relation:

$$W(i) = W(g_k) \otimes (g_k - i)$$

Where $A \otimes a$ denoted the set $\{x - a| \in A \text{ and } x > a\}$ for an integer set A and a scaler a. Thus, it is sufficient to construct $W(g_k)$ using the algorithm, and then derive all other elements of the base set using a simple shift.

Example 5 Consider a linear array which has link structure $\{1, 3, 7, 10\}$. In this example g_k is 10 and the window is of size 20. By applying the above algorithm, we can find $W(10)$, the solution which is forced by 10. $W(10)$ in this case turns out to be the set (10, 7, 4, 3, 1). The corresponding fault pattern is (2, 5, 6, 8, 9, 11, 13, 14, 17, 20) for that window. By using the property, $W(i)= W(g_k) \otimes (g_k - i)$, we can find other $W(i)$, for $i=1,2, \ldots, 9$. The complete base set solution is shown in Table I.

4. GENERATION OF CATASTROPHIC FAULT PATTERNS

In the previous section, the correspondence between fault patterns and solutions was established. It was found that every fault pattern could be completely characterized by the non-faulty integers in the first half of the window, called the solution. By simply enumerating the number of all possible solutions for a window of size $2g_k$ would mean finding all the fault pattern for the window. The derivation of the base set solutions was a major step towards the process of fault pattern enumeration. Although the base set does not contain all the solutions, it can be used to generate other solutions.

The following properties illustrate the operations that can be performed on the base set solutions to derive other solutions.

PROPERTY 2 If W is a solution. then $W \otimes c$ is a solution. This is the translation property. Starting with a solution and performing simple shifts, we can obtain several other solutions.

PROPERTY 3 If W_a and W_b are solutions then $W_a \cup W_b$ is a solution. That is, the set of solutions is closed under union.

TABLE I Complete base set solution for {1,3,7,10}

i	$W(i)$	$F[W(i)]$
10	(10,7,4,3,1)	(2,5,6,8,9,11,13,14,17,20)
9	(9,6,3,2)	(1,4,5,7,8,10,12,13,16,19)
8	(8,5,2,1)	(3,4,6,7,9,10,11,12,15,18)
7	(7,4,1)	(2,3,5,6,8,9,10,11,14,17)
6	(6,3)	(1,2,4,5,7,8,9,10,13,16)
5	(5,2)	(1,3,4,6,78,9,10,12,15)
4	(4,1)	(2,3,5,6,7,8,9,10,11,14)
3	(3)	(1,2,4,5,6,7,8,9,10,13)
2	(2)	(1,3,4,5,6,7,8,9,10,12)
1	(1)	(2,3,4,5,6,7,8,9,10,11)

Example 6 $W_a = \{1, 3, 4, 7, 10\}$ and $W_b = (1, 2, 5, 8)$ are two solutions for $G = \{1, 3, 7, 10\}$. Notice that $W_a \cup W_b = (1, 2, 3, 4, 5, 7, 8, 10)$ is also a solution for G.

PROPERTY 4 Every solution can be expressed as the union of elements of the base set.

Property 4 indicates that every solution can be expressed as the union of elements of the base set. The union of two or more elements in the base set does not necessarily produce an unique solution: it may be a solution that is already in the base set. Our objective is to count the number of distinct solutions. To avoid duplication and miscalculation in enumeration, we use a graph theoretical approach. The algorithm is outlined in the following steps:

1. Compute the base sets $\{W(i)\colon i = 1, 2, \ldots, g_k\}$. Recall that $W(i)$, the solution forced by i, also contains i.
2. Form a directed acyclic graph $\mathcal{G}(N, E)$ where the nodes N are labelled $1, \ldots, g_k$ and an edge $e_1 \rightarrow e_2$ belongs to E if for some $i \in \{1, \ldots, g_k\}$: $W(i) = (\ldots, e_1, e_2, \ldots)$.
3. Take the transitive closure of $\mathcal{G}(N, E)$, remove sense of orientation, and denote it by $\mathcal{G}^*(N, E)$.
4. Take the complement of $\mathcal{G}^*(N, E)$. The number of edges in $\mathcal{G}^*(N, E)$ gives all the missing and distinct solutions.

The total number of solutions, and hence the number of fault patterns, is the sum of the edges in $\mathcal{G}^*$, $E(\mathcal{G}^*)$, together with the base set solutions. Finding the exact value of $E(\mathcal{G}^*)$ is somewhat involved. $E(\mathcal{G}^*)$ depends entirely on the structure of the base set solution which is again determined by the set of links G. We, therefore, give an approximate range for it. By looking at step 2 of the algorithm, we may find two extreme cases for the structure of $\mathcal{G}$. That is, $\mathcal{G}$ can be a graph with g_k nodes and $g_k - 1$ edges in which case $E(\mathcal{G}^*) = 0$. Alternatively, $\mathcal{G}$ can be a forest with g_k nodes and no edges in which case $E(\mathcal{G}^*) = [g_k^2 - g_k]/2$. Therefore, the total number of solutions (hence, fault patterns) for a linear array with $G = \{g_1, \ldots, g_k\}$ lies somewhere between g_k and $[g_k^2 + g_k]/2$.

5. CONCLUSION

In this paper, the importance of fault distribution and the link redundancy in fault tolerance of linear systolic arrays was studied. It was shown that certain fault patterns involving isolated and block faults could exist within the array that would deter further restructuring of the array, irrespective of the amount of PE redundancy built into it. In order for any fault pattern to pose a threat to reconfigurability, it was necessary for the fault pattern to have a minimum number of faults and be located with a frame or window of some maximum length. Both the minimum number of faults and the maximum window size have been shown to be functions of the length of the longest bypass link. Above all, the two halves of the window must be symmetric. Given a link configuration, it is always possible to form a fault pattern which will be catastrophic. The fault pattern is certainly not unique. We showed a method for enumerating all possible fault patterns for a given link configuration. The method of constructing fault patterns for linear array can be readily applied for similar purpose in structures of higher dimensions and many other regular architectures [9].

References

[1] Chean, M. and Fortes, J. A. B. (Jan. 1990). A taxonomy of reconfiguration techniques for fault-tolerant processor arrays, *IEEE Computer*, **23**(1), pp. 55–69.

[2] Chen, C., Feng, A., Takada, Y., Kikuno, T. and Torii, K. (1990). Spare processor assignment for reconfiguration of fault-tolerant arrays, *Trans. of the IEICE*, **E-73**(8), pp. 1247–1256.

[3] Greene, J. and Gamal, A. (Oct. 1984). Configuration of VLSI arrays in the presence of defects, *Journal of the ACM*, **31**(4), pp. 694–717.

[4] Hosseini, S. H. (1989). On fault-tolerant structure, distributed fault-diagnosis, reconfiguration, and recovery of the array processors, *IEEE Trans. on Computers*, **C-38**(7), pp. 932–942.

[5] Kung, H. T. (Jan. 1982). Why systolic architectures, *IEEE Computer*, **15**(1), pp. 37–46

[6] Li, H. F., Jayakumar, R. and Lam, C. (1989). Restructuring for fault-tolerant systolic arrays, in *IEEE Trans. on Computers*, **C-38**(2), pp. 307–311.

[7] Lombardi, F., Sami, M. G. and Stefanelli, R. (Sept. 1989). Reconfiguration of VLSI arrays by covering, *IEEE Trans. on Computer-Aided Design*, **8**(9), pp. 952–965.

[8] Mangir, T. E. (June 1984). Sources of failures and yield improvement for VLSI and restructurable interconnects for RVLSI and WSI: Part I- Sources of failures and yield improvement for VLSI, *Proc. of IEEE*, **72**, pp. 690–708.

[9] Nayak, A. (1991). On reconfigurability of some regular architectures, Ph.D. Thesis, Dept. of Systems and Computer Engineering, Carleton University, Ottawa, Canada.

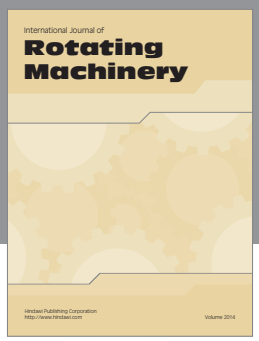
[10] Nayak, A., Santoro, N. and Tan, R. (1990). Fault-intolerance of reconfigurable systolic arrays, *in 20th Int'l Symp. on Fault-Tolerant Computers*, Newcastle upon Tyne, pp. 202–209.

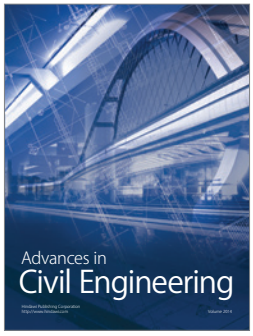
[11] Roychowdhury, V. P., Bruck, J. and Kailath, T. (1990). Efficient algorithms for reconfiguration in VLSI/WSI arrays, *IEEE Trans. on Computers*, **C-39**(4), pp. 480–489.

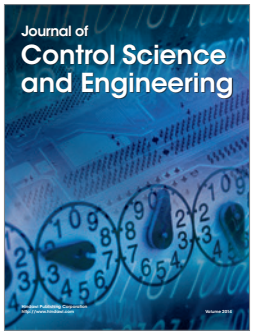
Authors' Biographies

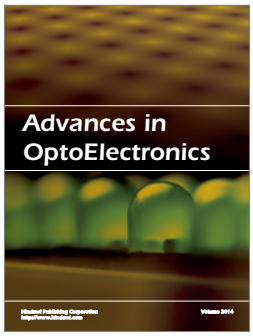
Vincenzo Acciaro is an assistant professor of Computer Science at University of Bari, Italy. He received the "Laurea in Scienze dell' Informazione" from University of Bari, the Master in Computer Science and Ph.D. from Carleton University, Ottawa, Canada in 1987, 1991 and 1995 respectively. His current research interests are in computational algebra.

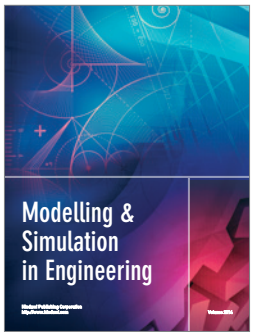
Amiya Nayak received his Ph.D. in Systems & Computer Engineering from Carleton University, Ottawa, Canada in 1991. He has been with Canadian Marconi Company, Ottawa since 1985. He is also an adjunct research professor at Carleton University. His research interests include fault-tolerant computing, testing of digital circuits and distributed processing.

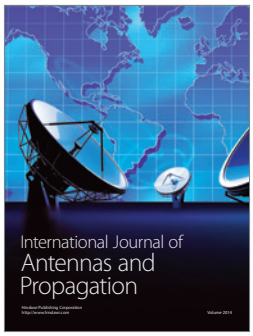


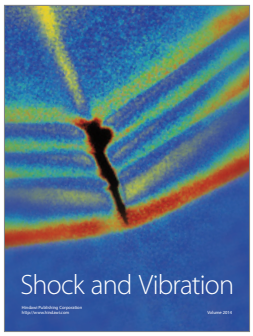


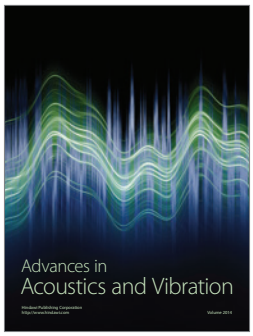












Hindawi

Submit your manuscripts at
http://www.hindawi.com