# RECOGNITION OF CATASTROPHIC FAULTS IN RECONFIGURABLE ARRAYS WITH ARBITRARY LINK REDUNDANCY

Amiya Nayak, Linda Pagli, Nicola Santoro

TR-204, MARCH 1992

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

# RECOGNITION OF CATASTROPHIC FAULTS IN RECONFIGURABLE ARRAYS WITH ARBITRARY LINK REDUNDANCY

Amiya Nayak†, Linda Pagli‡, Nicola Santoro†

† Center for Parallel & Distributed Computing, School of Computer Science, Carleton University, Ottawa, K1S 5B6.

‡ Dipartimento di Scienze dell'Informazione, University of Pisa, Corso Italia 40, 56100 Pisa, Italy.

**Abstract**

Fault tolerance through the incorporation of redundancy and reconfiguration is quite common. The distribution of faults can have severe impact on the effectiveness of any reconfiguration scheme; in fact, patterns of faults occurring at strategic locations may render an entire system unusable regardless of its component redundancy and of its reconfiguration capabilities. The characterization of such patterns is crucial for the identification and detection of such catastrophic events. A complete characterization was given for reconfigurable systolic arrays with 2-link redundancy; i.e., a bypass link of fixed length is provided to each element of the array in addition to the regular link.

In this paper, we study the more general case of arbitrary (but regular) link redundancy. In particular, we focus on the problem of deciding whether a pattern of $k$ faults is catastrophic for a k-link redundant system; i.e., in addition to the regular link of length $g_1 = 1$, each element of the array is provided with k-1 bypass links of length $g_2, g_3, \ldots, g_k$, respectively.

We study this problem and prove some fundamental properties which any catastrophic fault pattern must satisfy. We then show that these properties together constitute a necessary and sufficient condition for a fault pattern to be catastrophic for k-link redundant system. As a consequence, we derive a provably correct recognition algorithm whose worse-case time complexity is $O(kg_k)$; this also improves on the previous algorithm for k = 2.

**Index Terms:** Fault-tolerance, Systolic arrays, Catastrophic fault patterns, Testing.

# I. INTRODUCTION

Fault tolerance is the survival attribute of computer systems; when a system is able to recover automatically from internal faults without suffering an externally perceivable failure, the system is said to be fault-tolerant. Faults can occur in all systems at all levels. Therefore, a proper fault-tolerance mechanism must be in place to cope with possible failures. A common and practical approach for achieving fault-tolerance in VLSI-based regular architectures is by incorporating component redundancy and mechanisms for reconfiguration of the architecture. The redundant processing elements (PEs) are used to replace any faulty PE(s); the redundant links are used to bypass the faulty PEs and reach the redundant PEs used as a replacement. In the literature, many algorithms [1-15,19-26] have been proposed which take into account the built-in redundancy and reconfigure the system in the presence of faulty PEs and faulty links. The main objective of all the reconfiguration algorithms is to map faulty elements to spares (using bypass links) while preserving the high degree of regularity and locality of reference required by the system to perform correctly.

The effectiveness of using redundancy to increase fault tolerance clearly depends on both the amount of redundancy and the reconfiguration capability of the system. It does however depend also on the distribution of the faults in the system. In fact, faults occurring at strategic locations in a regular architecture may have catastrophic effect on the entire structure and cannot be overcome by any amount of clever design. For a given design, it is not difficult to identify a set of elements whose failure will have catastrophic consequence. For example, in a linear array of PEs 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. Similarly, the same array with $k-1$ bypass links $\{g_2, g_3, \ldots, g_k\}$ cannot tolerate $g_k$ (not $k$) PE faults if they occur in a block (or cluster). The probability of block faults of size $g_k$ or higher is relatively small; however, there exist many patterns (random distribution) of $g_k$ faults, not in a block, which can be fatal for the system. Therefore, the characterization of such fault patterns is obviously crucial for the identification, testing and detection of such catastrophic events.

The fault patterns that are catastrophic have been extensively studied for classes of regular architectures [16]. The knowledge about the catastrophic fault patterns can be used in many ways to improve reliability of regular systems. The knowledge about the catastrophic fault patterns can be applied to test for the likelihood of a catastrophe in regular systems. It is also possible to evaluate a design, using the characterization of catastrophic fault patterns, to verify if specific patterns of faults are catastrophic; should this be the case, any future design can be upgraded by incorporating appropriate redundancy structure into the design to minimize catastrophe. The patterns of faults can very well be the distribution of faults, frequently observed in the post-manufacturing phase or in an application domain.

In this paper, we are concerned with the development of efficient recognition schemes; that is, efficient mechanisms which automatically determine whether or not an observed/detected pattern of faults will have catastrophic consequences. The availability of such recognition schemes can have many practical consequences; e.g., they can be used after the fault detection/location phase to determine whether reconfiguration is possible, well ahead of time before the system is used in a critical operation. Similarly, the recognition schemes can be used when generating fault patterns to test for reconfigurability.

The problem of recognizing whether a fault pattern is catastrophic has been addressed only for specific cases. In particular, a $O(g^2)$ recognition algorithm was derived in [16,18] for 2−link redundant systems, where $g$ is the length of the bypass link.

In this paper, we study the more general case of recognizing catastrophic fault patterns in reconfigurable arrays with arbitrary link redundancy; i.e., $k > 1$. For these, we prove some fundamental properties which any catastrophic fault pattern must satisfy. We then show that these properties together constitute a necessary and sufficient condition for a fault pattern to be catastrophic for $k$−link redundant system. As a consequence, we derive a provably correct recognition algorithm whose worse-case time complexity is $O(kg_k)$; thus, we also improve on the previous algorithm for 2−link redundant systems.

The remaining of this paper is organized as follows. Basic concepts that provide basis for further analysis are introduced in Section II. In Section III, a representation for fault patterns

based on Boolean matrices is given, and the existing recognition algorithm is described. A special fault pattern, called the reference fault pattern, is introduced and its properties are descibed in Section IV. Necessary and sufficient conditions for a pattern to be catastrophic are given in Section V. An improved recognition scheme is presented in Section VI followed by a conclusion in Section VII.

## II. PRELIMINARIES

In this paper, we will focus on one-dimensional (or linear) arrays. The basic component of such an array is the processing element (PE) as shown in Figure 1. The peripheral PEs are responsible for I/O functions. The links can be either unidirectional or bidirectional. There are two kind of links in redundant arrays: *regular* or *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. Bypass links are shown in bold in Figure 1.

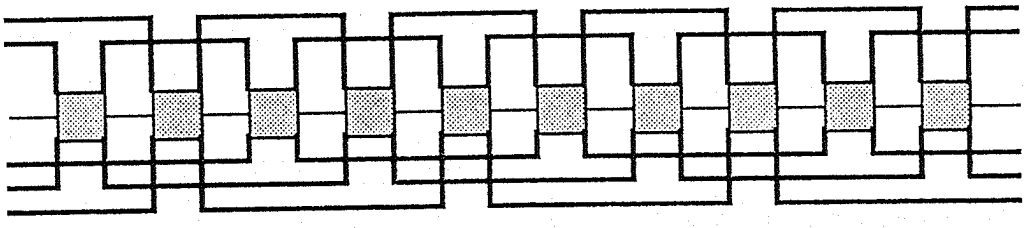


Figure 1: One dimensional array of processors

Let $A = \{p_0, p_2, \ldots, p_N\}$ denote a one-dimensional array of PEs, where each $p \in A$ represents a processing element and there exists a direct link between $p_i$ and $p_{i+1}, 0 \leq i < N$. Any link connecting $p_i$ and $p_j$ where $j > i+1$ is said to be a *bypass link*. The length of a bypass link, connecting $p_i$ and $p_j$, is the distance in the array between $p_i$ and $p_j$; i.e., $|j-i|$.

**Definition 1** *Given an integer $g \in [1, N]$ and an array $A$ of size $N$, $A$ is said to have link redundancy $g$, if for every $p_i \in A$ with $i \leq N - g$ there exists a link between $p_i$ and $p_{i+g}$; if $g > 1$, such a link will be called a bypass link.*

The above definition can be extended to a set of bypass links as follows:

**Definition 2** *The array $A$ has link redundancy $G = \{g_1, g_2, \ldots, g_k\}$ where $g_j < g_{j+1}$ and $g_j \in [1, N]$, if $A$ has link redundancy $g_1, g_2, \ldots, g_k$.*

In the following, it will be assumed that no other links exist in the array except the ones specified by $G$. Thus, $G$ totally defines the *link structure* of $A$, and $A$ will be called a *$k-$redundant system.* Notice that $g_1 = 1$ is the regular link, while all other $g_i$'s correspond to bypass links.

Given a linear array $A$ of size $N$, a *fault pattern* for $A$ is a set of integers $F = \{f_0, f_1, \ldots, f_m\}$ where $m \leq N, f_j < f_{j+1}$ and $f_j \in [0, N]$. An assignment of a fault pattern $F$ to $A$ means that for every $f \in F$, $p_f$ is faulty.

**Definition 3** *The width $W_F$ of a fault pattern $F$ is the number of PEs between and including the first and the last fault in $F$. That is, if $F = \{f_0, \ldots, f_m\}$ then $W_F = f_m - f_0 + 1$.*

**Definition 4** *A fault pattern $F$ is catastrophic for an array $A$ with link redundancy $G$ if the array cannot be reconfigured in the presence of such an assignment of faults.*

In other words, $F$ is a cut-set of the graph corresponding to $A$; that is, the removal of the faulty elements and their incident links will cause the array to become disconnected. A characterization of catastrophic fault patterns was given in [16,18]. It was shown that a catastrophic fault pattern for a link configuration $G = \{g_1, g_2, \ldots, g_k\}$ must have at least $g_k$ number of faults. Also, the width of a fault pattern must be bounded for the pattern to be catastrophic. Bounds were established on the width of the fault window $W_F$ for different link configurations. In this paper, we will consider *minimal* catastrophic fault patterns; that is, fault patterns which have exactly $g_k$ faulty PEs.

## III. MATRIX REPRESENTATION FOR FAULT PATTERNS AND THE EXISTING RECOGNITION ALGORITHM

We now introduce a representation for fault patterns based on Boolean matrices. This representation will be instrumental in establishing new properties of catastrophic fault patterns and deriving an efficient recognition algorithm. The purpose of this section is to define the representation and describe the existing recognition algorithm.

Consider an arbitrary fault pattern $F = \{f_0, f_1, \ldots, f_{g_k-1}\}$, consisting of $g_k$ faults for an arbitrary link configuration $G = \{g_1, g_2, \ldots, g_k\}$. Without loss of generality, assume that $f_0 = 0$. The links can be either unidirectional or bidirectional. We represent $F$ by a Boolean matrix $W$ of size $(W_F^+ \times g_k)$, where $W_F^+ = \lceil W_F/g_k \rceil$, defined as follows:

$$W[i,j] = \begin{cases} 1 & \text{if } ig_k + j \in F \\ 0 & \text{otherwise} \end{cases}$$

In the matrix representation, each $f_l \in F$ is mapped into $W[x_l, y_l]$ where $x_l = \lfloor f_l/g_k \rfloor$ and $y_l = f_l \bmod g_k$. Notice that $W[0,0] = 1$ which indicates the location of the first fault.

**Example 1:** Consider the fault pattern $F_1$ (shown in Figure 2) for an array of PEs with bidirectional links with link configuration $G = \{1, 10\}$; $\|F_1\| = 10$ and $W_F = 34$. The Boolean matrix representation of $F_1$ is shown Figure 3.

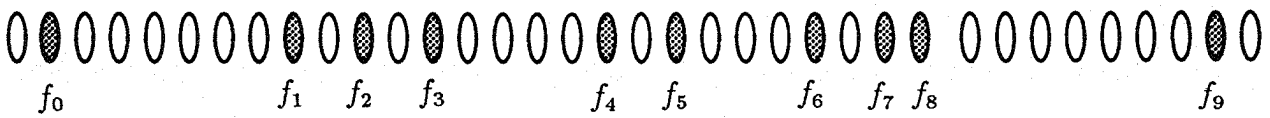



Figure 2 : A fault pattern $F_1$ for $G = \{1, 10\}$

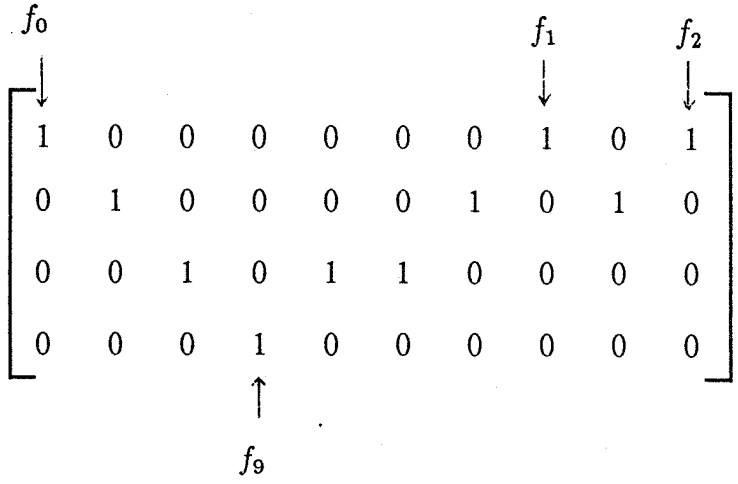



Figure 3 : The matrix representation for $F_1$ in Figure 2

Let $W$ be the matrix representation of a minimal fault pattern $F$. Notice that any minimal catastrophic fault pattern satisfies the necessary condition that $\forall j$, there is only one $i$ for which $W[i,j] = 1$. The *row coordinates* of $F$ is the ordered set $\{x_0, x_1, \ldots, x_{g_k-1}\}$ of the row indices of $W$ corresponding to the faults $f_i$ $(0 \leq i \leq g_k - 1)$. We now define the *interior, exterior*, and *border* elements in the matrix representation of a fault pattern.

**Definition 5** *Let $W[x_l, y_l]$ be the location of fault $f_l$. The location $W[i, y_l]$, with respect to $f_l$, is interior if $i < x_l$, border if $i = x_l$, and exterior if $i > x_l$.*

The definition of interior, border, and exterior can now be extended from element to regions as follows:

**Definition 6** *For a given fault pattern $F$, $I(F)$ (i.e., interior of $F$) is the set of all interior elements, $B(F)$ (i.e., border of $F$) is the set of all border elements, and $E(F)$ (i.e., exterior of $F$) is the set of all exterior elements.*

**Example 2:** Consider the fault pattern $F = \{0, 5, 9, 11, 14, 16, 18, 22, 23, 27\}$ with 10 faults in an array with the link configuration $G = \{1, 5, 10\}$ in which all links are bidirectional. The interior, border, and exterior elements are shown in Figure 4; the first and last row in Figure 4 correspond to elements in the array which are outside of $W_F$ and not part of $W$.

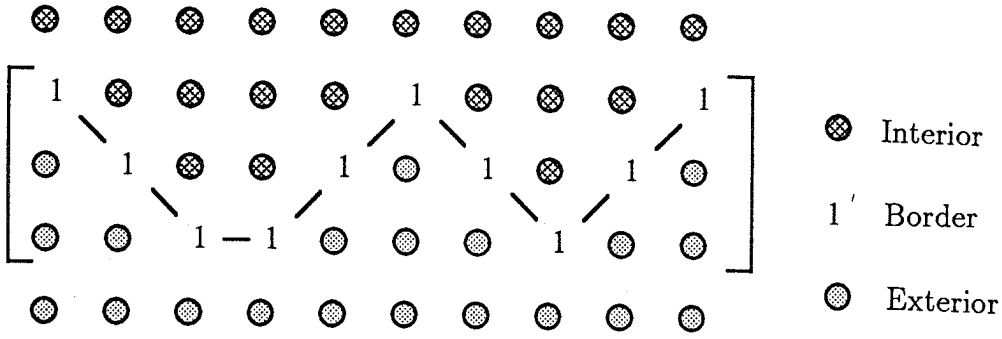



Figure 4 : *Interior*, *exterior* and *border* of a fault pattern.

Now with respect to the matrix representation of $F$, the definition for a catastrophic fault pattern (see Definition 4) can be rephrased in terms of interior and exterior regions as follows:

**Definition 7** *A fault pattern $F$ is catastrophic for an array $A$ with link redundancy $G$ if it is not possible to reach any exterior element from any interior element using the links in $G$.*

In the following, using the terminologies just introduced, we describe the existing algorithm for recognizing whether or not a fault pattern $F = \{f_0, f_1, \ldots, f_{g_k-1}\}$ is catastrophic for a link configuration $G = \{g_1, g_2, \ldots, g_k\}$.

**Algorithm 1:** Recognizing if a Fault Pattern is Catastrophic

Begin

TEST := TRUE;

*for* every element $x \in I(F)$ *do*

*for* every link $g \in G$ *do*

*if* $x + g \in E(F)$ *then* TEST := FALSE;

*endfor*

*endfor*

End.

This algorithm considers all elements in $I(F)$ and all links in $G = \{g_1, g_2, \ldots, g_k\}$ and verifies, for each interior element, whether it is possible to reach an exterior element using any link in $G$. The algorithm can obviously be rephrased so that it terminates as soon as TEST becomes FALSE. In any case, the complexity of the algorithm is bounded above by $\|I(F)\|k$, where $k$ is the number of links in $G$; the bound is exact if, for example, the pattern is catastrophic. Since $\|I(F)\| = O(g_k^2)$, the worst case complexity of the algorithm is $O(kg_k^2)$. This is basically the recognition algorithm used for 2−link redundant systems [16,18]; in this case, $k = 2$ and the complexity is $O(g_k^2)$.

The rest of this paper is dedicated to the formulation of a more efficient recognition algorithm. We are able to achieve this by exploiting several inherent properties, not previously studied, of the catastrophic fault patterns.

## IV. PROPERTIES OF THE REFERENCE FAULT PATTERN

In this section, we will consider a special fault pattern, called the *reference fault pattern*, for a link configuration and describe some of its properties.

The *area* of a fault pattern in its matrix representation can be defined as follows.

**Definition 8** *The area $A_F$ of a fault pattern $F$ is the number of interior and border elements; that is,*

$$A_F = \|I(F) \cup B(F)\| = \sum_{j=0}^{g_k-1} (x_j - 1).$$

Recall that the width $W_F$ (see Definition 3) of a fault pattern $F = \{f_0, f_1, \ldots, f_{g_k-1}\}$ is $W_F = f_{g_k-1} - f_0 + 1$. A reference fault pattern can now be defined in terms of its width and area as follows:

**Definition 9** *Given a link configuration $G$, a reference fault pattern (RFP) is a catastrophic fault pattern for $G$ which has largest width $W_F$ and maximum area $A_F$.*

Algorithms to construct *RFP*'s for arrays with unidirectional and bidirectional links, respectively, were given in [16,18]. In both cases, the algorithms construct a reference fault pattern $\mathcal{F}$ with time complexity $O(W_F + kg_k)$ and space complexity $O(W_F)$, where $k = \|G\|$ is the number of links. Recently, a more efficient algorithm for constructing the reference fault pattern has been presented in [17], and its complexity is $O(kg_k)$.

We will now establish some properties of the *RFP*'s using the matrix representation described in Section III. Since any catastrophic fault pattern is invariant with respect to translation [16] (i.e., if $F$ is catastrophic then $F + c$ is also catastrophic for any arbitrary integer $c$ and vice versa), we can assume without loss of generality that $f_0 = 0$.

Let $\mathcal{F}$ be a reference fault pattern. By definition of reference fault pattern, $W_{\mathcal{F}}$ is maximal and $A_{\mathcal{F}}$ is maximal.

Consider two fault patterns $F_\alpha = \{f'_0, f'_1, \ldots, f'_{g_k-1}\}$ and $F_\beta = \{f''_0, f''_1, \ldots, f''_{g_k-1}\}$ for a given link configuration $G$. We define the concatenation of $F_\alpha$ and $F_\beta$ as follows:

**Definition 10** *Let* $\{x'_0, x'_1, \ldots, x'_{g_k-1}\}$ *and* $\{x''_0, x''_1, \ldots, x''_{g_k-1}\}$ *be the row coordinates of* $F_\alpha$ *and* $F_\beta$ *respectively in their respective matrix representation. The concatenation of* $F_\alpha$ *and* $F_\beta$ *(denoted by* $F_\alpha \| F_\beta$*) is a fault pattern* $F$ *whose row coordinates are* $\{x_0, x_1, \ldots, x_{g_k-1}\}$*, where* $x_i = \{max(x'_i, x''_i)\}$ *for* $0 \leq i \leq g_k - 1$.

An interesting property of the concatenation operation is the following:

**Property 1** *Let* $F_\alpha$ *and* $F_\beta$ *be catastrophic for* $G$*. Then, their concatenation* $F_\alpha \| F_\beta$ *is also catastrophic for* $G$.

**Proof:** Let $F = F_\alpha \| F_\beta$. We must show that no exterior element of $F$ is reachable from any interior element of $F$. $F = F_\alpha \| F_\beta$ implies $I(F_\alpha) \cup B(F_\alpha) \subseteq I(F) \cup B(F)$ and $I(F_\beta) \cup B(F_\beta) \subseteq I(F) \cup B(F)$. Let $x$ be any arbitrary element in $I(F)$ and $g$ be an arbitrary link in $G$. We will now show that $x + g \in I(F) \cup B(F)$.

**Case 1:** $x \in I(F_\alpha)$. Since $F_\alpha$ is catastrophic, $x \in I(F_\alpha)$ implies $x + g \in I(F_\alpha) \cup B(F_\alpha) \subseteq I(F) \cup B(F)$.

**Case 2:** $x \in I(F_\beta)$. Similarly, using the fact that $F_\beta$ is catastrophic, $x \in I(F_\beta)$ implies $x + g \in I(F_\beta) \cup B(F_\beta) \subseteq I(F) \cup B(F)$.

**Case 3:** $x \in B(F_\alpha)$. In this case, since $x \in I(F)$, $x \in I(F_\beta)$. Therefore, by Case 2, $x + g \in I(F_\beta) \cup B(F_\beta) \subseteq I(F) \cup B(F)$.

**Case 4:** $x \in B(F_\beta)$. $x \in I(F)$ implies that $x \in I(F_\alpha)$. Therefore, by Case 1, $x + g \in I(F_\alpha) \cup B(F_\alpha) \subseteq I(F) \cup B(F)$.

Since both $x$ and $g$ are arbitrary, it follows that it is not possible to reach any exterior element from any interior element in $F_\alpha \| F_\beta$ using any link in $G$. Hence, $F_\alpha \| F_\beta$ is catastrophic for $G$. □

We use this property to prove that, for a link configuration $G$, there is one and only one reference fault pattern.

**Property 2** *For any link configuration $G$, the reference fault pattern is unique.*

**Proof:** By contradiction, let $F_\alpha \neq F_\beta$ be two reference fault patterns for $G$. By definition, $A_{F_\alpha} = A_{F_\beta} = A_{max}$. By Property 1, $F = F_\alpha \| F_\beta$ is also catastrophic for $G$; on the other hand, $A_F > A_{F_\alpha} = A_{max}$, contradicting the definition of maximal area. □

This property implies that $\mathcal{F}$, constructed by the algorithms in [16-18], is the unique reference fault pattern. We are now in a position to prove the necessary and sufficient conditions for a fault pattern to be catastrophic for a given link configuration.

## IV. NECESSARY & SUFFICIENT CONDITIONS FOR CATASTROPHE

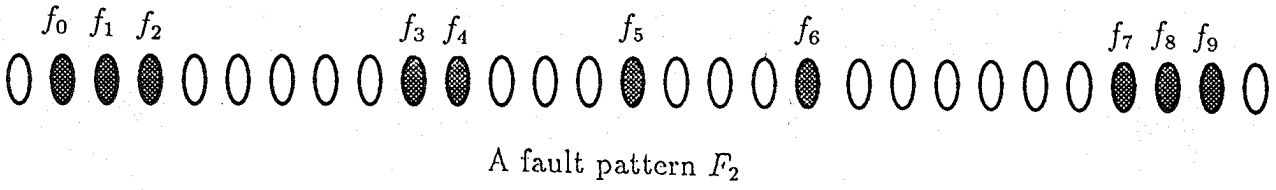
The first necessary condition establishes an important relationship between a fault pattern $F$ and the reference fault pattern $\mathcal{F}$.

**Definition 11** *For any two fault patterns $F_\alpha$ and $F_\beta$, $F_\alpha$ and $F_\beta$ cross if $I(F_\alpha) \not\subseteq I(F_\beta)$ and $I(F_\beta) \not\subseteq I(F_\alpha)$.*

**Lemma 1** *Given $G$, let $\mathcal{F}$ be the reference fault pattern and $F$ be any fault pattern for $G$. If $F$ and $\mathcal{F}$ cross, then $F$ is not catastrophic for $G$.*

**Proof:** We will prove the lemma by contradiction. Suppose $F$ is catastrophic. Since $F$ crosses the reference fault pattern $\mathcal{F}$, it implies that $I(F) \not\subseteq I(\mathcal{F})$. Consider the new fault pattern, $\mathcal{F} \| F$, which is the concatenation of $F$ and $\mathcal{F}$. By Property 1, $\mathcal{F} \| F$ is catastrophic. Furthermore, $A_{\mathcal{F}\|F} > A_{\mathcal{F}}$, contradicting the fact that the reference fault pattern $\mathcal{F}$ has the largest area. Therefore, the lemma follows. □

**Example 3:** Figure 5 shows the matrix representation of the reference fault pattern $\mathcal{F}$ and a fault pattern $F_2$ with 10 faults for $G = \{1, 5, 10\}$. The links in this case are bidirectional. The solid line and dashed line indicate the border of $\mathcal{F}$ and $F_2$ respectively. Notice that $F_2$ crosses $\mathcal{F}$ and, therefore, is not catastrophic. The escape path is shown in the figure.




A fault pattern $F_2$

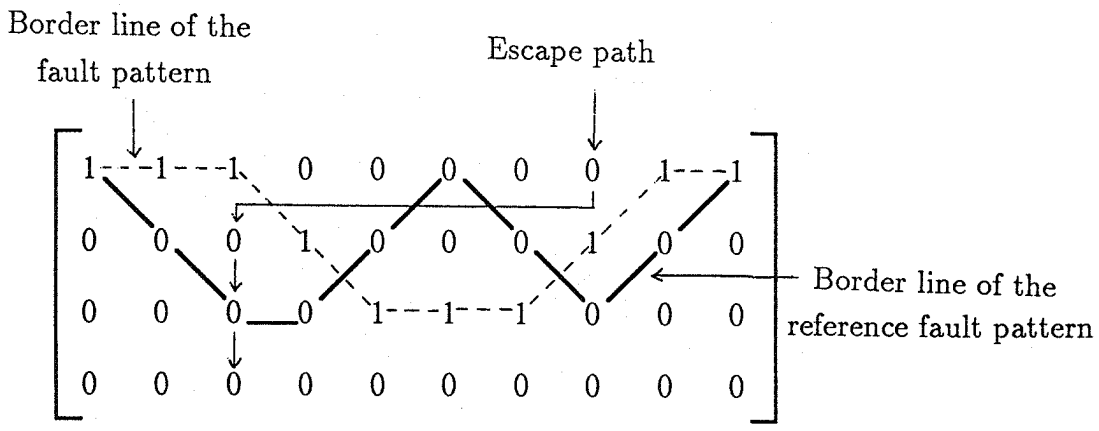
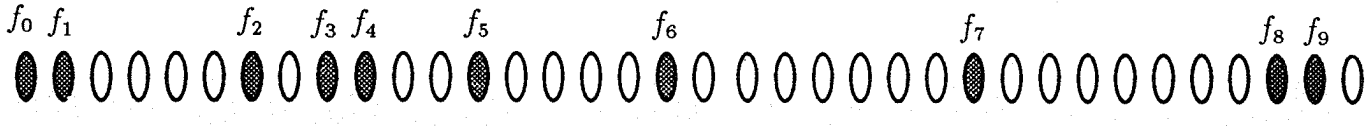



Figure 5 : A fault pattern $F_2$ which crosses the reference fault pattern for $G = \{1, 5, 10\}$

Lemma 1 expresses a necessary condition for a fault pattern to be catastrophic. However, not crossing $\mathcal{F}$ is not sufficient for a fault pattern to be catastrophic. The following example illustrates such a case.

**Example 4:** Figure 6 shows the matrix representation of $\mathcal{F}$ and a fault pattern $F_3$ with 10 faults for $G = \{1, 10\}$ when the links are bidirectional. The solid line and dashed line indicate the border of $\mathcal{F}$ and $F_3$ respectively. Note that although $F_3$ does not cross $\mathcal{F}$, it is still not catastrophic. The escape path is shown in the figure.




A fault pattern $F_3$

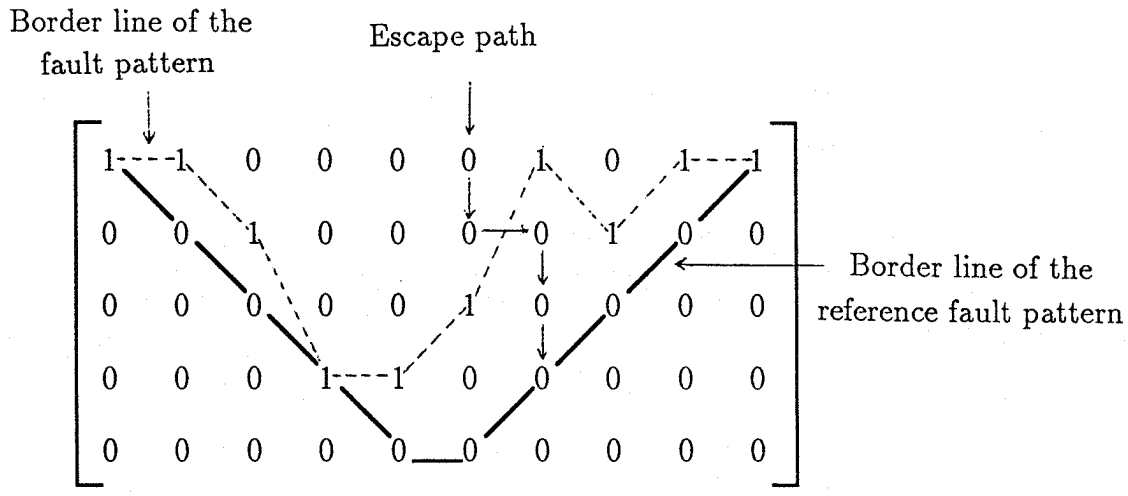



Figure 6 : A fault pattern $F_3$ which does not cross the reference fault pattern for $G = \{1, 10\}$

There exist additional conditions which a fault pattern must satisfy to be catastrophic. Such conditions are expressed in this section; based on these conditions, the improved recognition algorithm is presented.

**Lemma 2** *Let the links be unidirectional. If $F$ does not satisfy the following property then $F$ is not catastrophic for $G$: for any column $y_i$ $(0 \leq y_i \leq g_k - 1)$ in $W$ and for any link $g \in G = \{g_1, g_2, \ldots, g_k\}$*

$$x_i \leq \begin{cases} x_{i+g} + 1 & \text{if } i + g \leq g_k - 1 \\ x_j & \text{otherwise} \end{cases}$$

*where $j = (i + g) \bmod g_k$.*

**Proof:** By contradiction, let $F$ be catastrophic for $G$ and there exist $i$ and $g$ such that the property does not hold. Consider first the case where $i + g \leq g_k - 1$ (see Figure 7a). If the property does not hold then $x_i > x_{i+g} + 1$. The element in position $(x_i - 1, y_i)$ is an

interior one; on the other hand, the element in position $(x_i - 1, y_{i+g})$, which is reachable from $(x_i - 1, y_i)$ using link $g$, is exterior, contradicting the fact that the pattern is catastrophic. Consider now the case $i + g > g_k - 1$ (see Figure 7b). If the property does not hold then $x_j \leq x_i - 1$ where $j = (i+g) \bmod g_k$. In this case, the interior element in position $(x_i - 1, y_i)$ can reach the exterior element in position $(x_i, y_j)$, contradicting the fact that the pattern is catastrophic. □

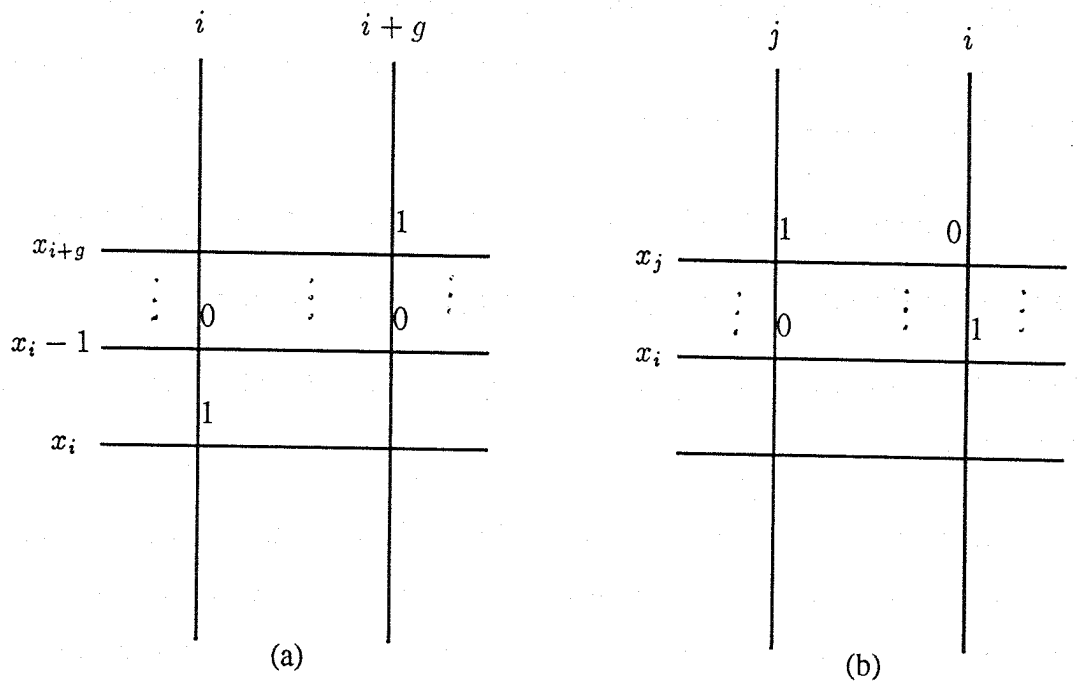



Figure 7 : Illustration of two cases for Lemma 2.

Lemma 2 can be extended to the case of bidirectional links as follows:

**Lemma 3** *For bidirectional links, if $F$ does not satisfy the following property then $F$ is not catastrophic for $G$: for any column $y_i$ $(0 \leq y_i \leq g_k - 1)$ in $W$ and for any link $g \in G = \{g_1, g_2, \ldots, g_k\}$*

$$1) \quad x_i \leq \begin{cases} x_{i+g} + 1 & \text{if } i + g \leq g_k - 1 \\ x_j & \text{otherwise} \end{cases}$$

$$2) \quad x_i \leq \begin{cases} x_{i-g} + 1 & \text{if } i - g \geq 0 \\ x_{\lfloor i+g_k-g \rfloor} + 2 & \text{otherwise} \end{cases}$$

*where $j = (i + g) \bmod g_k$.*

**Proof:** Condition 1) is the same as the condition in Lemma 2, and it must hold when links are used in the forward direction. Condition 2) refers to the case when the links can be used in backward direction. In the case $i - g \geq 0$, the elements in column $i$ can reach the elements in column $i - g$, lying in the same line; hence, the condition is same as in case 1). For $i - g < 0$, the elements in column $i$ can reach elements in column $i + g_k - g$, lying one row above. Using the same reasoning as in Lemma 2, the properties can be easily proved. □

The above lemmas state necessary conditions for a fault pattern to be catastrophic. We will now show that the combination of the conditions expressed by Lemmas 1, 2, and 3 constitute a necessary and sufficient condition.

**Theorem 1** *A fault pattern $F$ is catastrophic for a link configuration $G$ if and only if*

*i) it does not cross the reference fault pattern corresponding to $G$, and*

*ii) it satisfies Lemma 2 in the case of unidirectional links and Lemma 3 in case of bidirectional links.*

**Proof:** In this proof, we will consider the links to be unidirectional. The proof is similar for the case of bidirectional links. The necessity part has already been shown in Lemmas 1 and 2. To prove the theorem, we must to show that if $F$ does not cross the reference fault pattern and satisfies Lemma 2 then the pattern is catastrophic. Consider an arbitrary column $i$ in $W$ and a link $g$ in $G$. Let $i + g \geq g_k - 1$. Since the property of Lemma 2 holds, $x_{i+g} + 1 \geq x_i$. Thus, every interior element in column $i$ reaches, using link $g$, an element in column $i + g$ which is either an interior or a border element. Let $i + g > g_k - 1$. Since the property in Lemma 2 holds, $x_j \geq x_i$ where $j = (i + g) \bmod g_k$. Thus, every interior element in column $i$ reaches, using link $g$, the element in column $i + g$ which is either interior or a border.

Since $g$ is arbitrary, it follows that interior elements in column $i$ can only reach elements which are either interior or border. Since $i$ is arbitrary, the proof is completed. □

## VI. AN IMPROVED RECOGNITION STRATEGY

In this section, we use the preceding results to construct an efficient recognition algorithm. In particular, the algorithm will verify whether the necessary and sufficient conditions expressed by Theorem 1 are met. The algorithm actually includes a pre-testing phase, ensuring that the width and area of $F$ are not greater than the ones of the reference fault pattern $\mathcal{F}$; recall (from Definition 9) that $W_{\mathcal{F}}$ and $A_{\mathcal{F}}$ are maximal.

**Algorithm 2:** Recognizing if $F$ is catastrophic for $G$

```
Begin
    TEST := True;
    Test for violation of maximal area and width;
    if TEST then
        Test for crossing;
        if TEST then Test for property;
    endif
End.
```

**Test for violation of maximal area and width** (Pre-Testing)

```
Begin
    if W_F < W_F or A_F < A_F then
        TEST := false
    endif
End;
```

**Test for Crossing**

```
Begin
    Let {x_i} and {x̄_i} be the row coordinates of F and F, respectively.
    i := 0;
    repeat
        if x_i > x̄_i then
            TEST := False
        endif
        i := i + 1;
    until i > g_k or not(TEST)
End;
```

**Test for Property**

```
Begin
    i := 0;
    repeat
        j := 1;
        repeat
            if i + g_j ≤ g_k − 1 then x_p := x_{i+g_j} + 1 else x_p := x_{(i+g_j) mod g_k};
            if i − g_j ≥ 0 then x_q := x_{i−g_j} + 1 else x_q := x_{⌊i+g_k−g_j⌋} + 2;
            Case link orientation of
            unidirectional:
                if x_i > x_p then
                    TEST := False
                endif
            bidirectional:
                if (x_i > x_p) and (x_i > x_q) then
                    TEST := False
                endif
            endcase
            j := j + 1;
        until j > k or not(TEST)
        i := i + 1;
    until i > g_k or not(TEST)
End;
```

The major steps of Algorithm 2 are: Test for Crossing and Test for Property. Test for Crossing requires only the determination of the maximal row coordinate of $F$ and $\mathcal{F}$; thus, it can be done in time $O(g_k)$. For each row coordinate and for each link, Test for Property requires either one or two tests depending on whether the links are unidirectional or bidirectional, respectively; hence, the entire process can be completed in time $O(kg_k)$. It is assumed that $\mathcal{F}$ is already available; should this not be the case, it can be easily constructed in time $O(kg_k)$ using the algorithm of [17].

Notice that the complexity of this algorithm represents an improvement on the $O(g_k^2)$ complexity of the existing algorithm for 2−link redundant systems (i.e., $G = \{1, g_k\}$ and $k = 2$); in fact, in this case, the proposed algorithm requires only $O(g_k)$ time.

Finally, observe that it is possible, for some $F$, that $\|I(F)\| < g_k$. Should this be the case, Algorithm 1, described in Section III, becomes more efficient than Algorithm 2. Since the value $\|I(F)\|$ can be computed in $O(g_k)$ time, we can integrate the two techniques obtaining a recognition algorithm which has an overall time complexity $O(g_k + min\{\|I(F)\|, g_k\}k)$.

## VII. APPLICATIONS AND CONCLUSIONS

Regular systems are being designed with massive redundancy built into them. These systems also make use of the redundancy to reconfigue in the event of failure in one or more components; normally, a reconfiguration process is triggered as soon as a fault is detected. The success of the reconfiguration process depends mainly on two factors: the availability of redundant components (level of redundancy) and the distribution of component faults. It is possible to provide a large number of redundant components with the current technology. With the incorporation of massive redundancy into a system, comes the increased likelihood of component failures. The reconfiguration process will now encounter not only more faults but also a variety of fault patterns. This raises the following questions:

- how effective is the reconfiguration process?
- should the device or system be used in an application which cannot afford reconfiguration failure?

The results of the paper provide some answers to these questions. The proposed scheme can be used to determine the likelihood of a catastrophe in the system or device when some of its component fail; that is, the scheme allows the designer to recognize efficiently and effectively if the occurence of specific patterns of faults will pose a problem and cannot be reconfigured. No such other mechanism exists to our knowledge. To be able to recognize such patterns is useful not only to test the effectiveness of the employed reconfiguration scheme but also to prevent a total system shutdown.

The results in this paper provide a set of tools which can be employed in

1. assessing the fault tolerance effectiveness of a design; this can be done by specifying the minimum number of faults which the design cannot be guaranteed to withstand,

2. testing whether a design meets the specified fault tolerance requirements; this can be achieved by comparing the requirements with the ones derived using the properties of the catastrophic fault patterns, and
3. determining redundancy requirement for the designer to meet a desired level of fault tolerance; this can be done by determining the minimal link configuration for which no catastrophic fault patterns exist below the specified amount of failure.

Furthermore, the results presented here can help to usefully incorporate knowledge of the application field into the design process as feedbacks to the designer. In particular, knowledge of the type and distribution of faults occurring in the application field can be used to determine for which designs those patterns are catastrophic; thus, the designer can remove those designs from further consideration (even though, without that knowledge, they might have been viable choices).

The technical details of this paper can be summarized as follows. We have studied the problem of recognizing whether a pattern of $k$ faults is catastrophic for a k-link redundant system; i.e., in addition to the regular link $g_1 = 1$, each element of the array is provided with k-1 bypass links of length $g_2, g_3, \ldots, g_k$, respectively. We have proved some fundamental properties which any catastrophic fault pattern must satisfy. We have shown that these properties together constitute a necessary and sufficient condition for a fault pattern to be catastrophic for k-link redundant system. As a consequence, we have derived a provably correct recognition algorithm whose worse-case time complexity is $O(kg_k)$; thus, we have improved the previous algorithm for k = 2.

ACKNOWLEDGEMENT

This work was supported in part by Natural Sciences and Engineering Research Council of Canada under Operating Grant A2415 and in part by Progetto Finalizzato Sistemi Informatici Calcolo Parallelo.

## REFERENCES


[1] J. A. Abraham and K. Fuchs, "Fault and Error Models for VLSI," *Proc. of the IEEE*, Vol. 74, No. 5, May 1986, pp. 639-654.

[2] K. P. Belkhale and P. Banerjee, "Reconfiguration Strategies in VLSI Processor Arrays," in *Proc. Int'l Conf. on Computer Design*, 1988, pp. 418-421.

[3] M. Chean and J. A. B. Fortes, "A Texanomy of Reconfiguration Techniques for Fault-Tolerant Processor Arrays," in *IEEE Computer*, Vol. 23, No. 1, Jan. 1990, pp. 55-69.

[4] F. Distante, F. Lombardi, and D. Sciuto, "Array Partitioning: A Methodology for Reconfigurability and Reconfiguration Problems," in *Proc. Int'l Conf. on Computer Design*, 1988, pp. 564-567.

[5] J. A. B. Fortes and C. S. Raghavendra, "Dynamically Reconfigurable Fault-Tolerant Array Processors," in *Proc. 14th Int'l Conf. on Fault-Tolerant Computers*, June 1984, pp. 386-392.

[6] P. Franzon, "Interconnect Strategies for Fault Tolerant 2D VLSI Arrays," in *Proc. Int'l Conf. on Computer Design*, 1986, pp. 230-233.

[7] J. W. Greene and A. E. Gamal, "Configuration of VLSI Arrays in the Presence of Defects," *Journal of the ACM*, Vol. 31, No. 4, 1984, pp. 694-717.

[8] S. H. Hosseini, "On Fault-Tolerant Structure, Distributed Fault-Diagnosis, Reconfiguration, and Recovery of the Array Processors," *IEEE Trans. on Computers*, Vol. C-38, No. 7, July 1989, pp. 932-942.

[9] J. H. Kim and S. M. Reddy, "On the Design of Fault-Tolerant Two-Dimensional Systolic Arrays for Yield Enhancement," *IEEE Trans. on Computers*, Vol. C-38, No. 4, April 1989, pp. 515-525.

[10] I. Koren and D. K. Pradhan, "Introducing Redundancy into VLSI Designs for Yield and Performance Enhancement," in *Proc. 15th Int'l Conf. on Fault-Tolerant Computers*, 1985, pp. 330-335.

[11] H. T. Kung and M. Lam, "Fault-Tolerant VLSI Systolic Arrays and Two-Level Pipelining," *Journal of Parallel and Distributed Processing*, Aug. 1984, pp. 32-63.

[12] S. Kuo and W. K. Fuchs, "Efficient Spare Allocation for Reconfigurable Arrays," *IEEE Design and Test*, Feb. 1987, pp. 24-31.

[13] T. Leighton and C. E. Leiserson, "Wafer-Scale Integration of Systolic Arrays," *IEEE Trans. on Computers*, Vol. C-34, No. 5, May 1985, pp. 448-461.

[14] H. F. Li, R. Jayakumar, and C. Lam, "Restructuring for Fault-Tolerant Systolic Arrays," *IEEE Trans. on Computers*, Vol. C-38, No. 2, 1989, pp. 307-311.

[15] T. E. Mangir and C. S. Raghavendra, "Issues in the Implementation of Fault-Tolerant VLSI and WSI Systems," in *Proc. Int'l Conf. on Computer Design*, 1984, pp. 95-100.

[16] A. Nayak "On Reconfigurability of some Regular Architectures," Ph.D Thesis, Dept. Systems & Computer Engineering, Carleton University, Ottawa, Canada, 1991.

[17] A. Nayak, L. Pagli and N. Santoro, "Efficient Construction of Catastrophic Patterns for VLSI Reconfigurable Arrays," Technical Report (TR-202), School of Computer Science, Carleton University, Ottawa, Canada, 1992.

[18] A. Nayak, N. Santoro and R. Tan, "Fault-Intolerance of Reconfigurable Systolic Arrays," in *20th Int'l Symp. on Fault-Tolerant Computers*, Newcastle upon Tyne, 1990, pp. 202-209.

[19] R. Negrini, M. G. Sami, and R. Stefanelli, "Fault-Tolerance Techniques for Array Structures used in Supercomputing," *IEEE Computer*, Vol. 19, No. 2, 1986, pp. 78-87.

[20] R. Negrini and R. Stefanelli, "Algorithms for Self-Reconfiguration of Wafer-Scale Regular Arrays," in *Proc. Int'l Conf. on Circuits and Systems*, 1985, pp. 190-196.

[21] S. P. Popli and M. A. Bayoumi, "Fault Diagnosis and Reconfiguration for Reliable VLSI Arrays," in *Proc. Conf. on Computers and Communications*, Phoenix, 1988, pp. 69-73.

[22] D. K. Pradhan, Ed., *Fault-Tolerant Computing, Theory and Techniques*, Vol. 1 and 2, Englewood Cliff, NJ: Prentice-Hall, 1986.

[23] A. L. Rosenberg, "The Diogenes Approach to Testable Fault-Tolerant Arrays of Processors," *IEEE Trans. on Computers*, Vol. C-32, No. 10, Oct. 1983, pp.902-910.

[24] V. P. Roychowdhury, J. Bruck and T. Kailath, "Efficient Algorithms for Reconfiguration in VLSI/WSI Arrays," *IEEE Trans. on Computers*, Vol. C-39, No. 4, 1990, pp. 480-489.

[25] L. A. Shombert and D. P. Siewiorek, "Using Redundancy for Concurrent Testing and Repairing of Systolic Arrays," in *Proc. 17th Int'l Conf. on Fault-Tolerant Computers*, 1987, pp. 244-249.

[26] H. Y. Youn and A. D. Singh, "A Highly Efficient Design for Reconfiguring the Processor Array in VLSI," in *Proc. Int'l Conf. on Parallel Processing*, 1988, pp. 375-382.

## School of Computer Science, Carleton University
## Recent Technical Reports

**TR-163** **Finding a Central Link Segment of a Simple Polygon in O(N Log N) Time**
L.G. Alexandrov, H.N. Djidjev, J.-R. Sack, October 1989

**TR-164** **A Survey of Algorithms for Handling Permutation Groups**
M.D. Atkinson, January 1990

**TR-165** **Key Exchange Using Chebychev Polynomials**
M.D. Atkinson and Vincenzo Acciaro, January 1990

**TR-166** **Efficient Concurrency Control Protocols for B-tree Indexes**
Ekow J. Otoo, January 1990

**TR-167** **A Hierarchical Stochastic Automaton Solution to the Object Partitioning Problem**
B.J. Oommen, January 1990

**TR-168** **Adaptive List Organizing for Non-stationary Query Distributions. Part I: The Move-to-Front Rule**
R.S. Valiveti and B.J. Oommen, January 1990

**TR-169** **Trade-Offs in Non-Reversing Diameter**
Hans L. Bodlaender, Gerard Tel and Nicola Santoro, February 1990

**TR-170** **A Massively Parallel Knowledge-Base Server using a Hypercube Multiprocessor**
Frank Dehne, Afonso Ferreira and Andrew Rau-Chaplin, April 1990

**TR-171** **Parallel Processing of Quad Trees on the Hypercube (and PRAM)**
Frank Dehne, Afonso Ferreira and Andrew Rau-Chaplin, April 1990

**TR-172** **A Note on the Load Balancing Problem for Coarse Grained Hypercube Dictionary Machines**
Frank Dehne and Michel Gastaldo, May 1990

**TR-173** **Self-Organizing Doubly-Linked Lists**
R.S. Valiveti and B.J. Oommen, May 1990

**TR-174** **A Presortedness Metric for Ensembles of Data Sequences**
R.S. Valiveti and B.J. Oommen, May 1990

**TR-175** **Separation of Graphs of Bounded Genus**
Ljudmil G. Aleksandrov and Hristo N. Djidjev, May 1990

**TR-176** **Edge Separators of Planar and Outerplanar Graphs with Applications**
Krzystof Diks, Hristo N. Djidjev, Ondrej Sykora and Imrich Vrto, May 1990

**TR-177** **Representing Partial Orders by Polygons and Circles in the Plane**
Jeffrey B. Sidney and Stuart J. Sidney, July 1990

**TR-178** **Determining Stochastic Dependence for Normally Distributed Vectors Using the Chi-squared Metric**
R.S. Valiveti and B.J. Oommen, July 1990

**TR-179** **Parallel Algorithms for Determining K-width-Connectivity in Binary Images**
Frank Dehne and Susanne E. Hambrusch, September 1990

**TR-180** **A Workbench for Computational Geometry (WOCG)**
P. Epstein, A. Knight, J. May, T. Nguyen, and J.-R. Sack, September 1990

**TR-181** **Adaptive Linear List Reorganization under a Generalized Query System**
R.S. Valiveti, B.J. Oommen and J.R. Zgierski, October 1990

**TR-182** **Breaking Substitution Cyphers using Stochastic Automata**
B.J. Oommen and J.R. Zgierski, October 1990

**TR-183** **A New Algorithm for Testing the Regularity of a Permutation Group**
V. Acciaro and M.D. Atkinson, November 1990

**TR-184** **Generating Binary Trees at Random**
M.D. Atkinson and J.-R. Sack, December 1990

**TR-185** **Uniform Generation of Combinatorial Objects in Parallel**
M.D. Atkinson and J.-R. Sack, January 1991

**TR-186** **Reduced Constants for Simple Cycle Graph Separation**
Hristo N. Djidjev and Shankar M. Venkatesan, February 1991

**TR-187** **Multisearch Techniques for Implementing Data Structures on a Mesh-Connected Computer**
Mikhail J. Atallah, Frank Dehne, Russ Miller, Andrew Rau-Chaplin, and Jyh-Jong Tsay, February 1991

**TR-188** **Generating and Sorting Jordan Sequences**
Out of print Alan Knight and Jörg-Rüdiger Sack, March 1991

**TR-189** **Probabilistic Estimation of Damage from Fire Spread**
Charles C. Colbourn, Louis D. Nel, T.B. Boffey and D.F. Yates, April 1991

**TR-190** **Coordinators: A Mechanism for Monitoring and Controlling Interactions Between Groups of Objects**
Wilf R. LaLonde, Paul White, and Kevin McGuire, April 1991

**TR-191** **Towards Decomposable, Reusable Smalltalk Windows**
Kevin McGuire, Paul White, and Wilf R. LaLonde, April 1991

**TR-192** **PARASOL: A Simulator for Distributed and/or Parallel Systems**
John E. Neilson, May 1991

**TR-193** **Realizing a Spatial Topological Data Model in a Relational Database Management System**
Ekow J. Otoo and M.M. Allam, August 1991

**TR-194** **String Editing with Substitution, Insertion, Deletion, Squashing and Expansion Operations**
B John Oommen, September 1991

**TR-195** **The Expressiveness of Silence: Optimal Algorithms for Synchronous Communication of Information**
Una-May O'Reilly and Nicola Santoro, October 1991

**TR-196** **Lights, Walls and Bricks**
J. Czyzowicz, E. Rivera-Campo, N. Santoro, J. Urrutia and J. Zaks, October 1991

**TR-197** **A Brief Survey of Art Gallery Problems in Integer Lattice Systems**
Evangelos Kranakis and Michel Pocchiola, November 1991

**TR-198** **On Reconfigurability of Systolic Arrays**
Amiya Nayak, Nicola Santoro, and Richard Tan, November 1991

**TR-199** **Constrained Tree Editing**
B. John Oommen and William Lee, December 1991

**TR-200** **Industry and Academic Links in Local Economic Development: A Tale of Two Cities**
Helen Lawton Smith and Michael Atkinson, January 1992

**TR-201** **Computational Geometry on Analog Neural Circuits**
Frank Dehne, Boris Flach, Jörg-Rüdiger Sack, Natana Valiveti, January 1992

**TR-202** **Efficient Construction of Catastrophic Patterns for VLSI Reconfigurable Arrays**
Amiya Nayak, Linda Pagli, Nicola Santoro, February 1992

**TR-203** **Numeric Similarity and Dissimilarity Measures Between Two Trees**
B. John Oommen and William Lee, February 1992