Reconfiguring Processor Arrays Using Multiple-Track Models: The 3-Track-1-Spare-Approach

Theodora A. Varvarigou, Vwani P. Roychowdhury, and Thomas Kailath, Fellow, IEEE

Abstract—We present new results on systematic procedures for reconfiguring processor arrays in the presence of faulty processors. In particular, we consider models that use multiple tracks along every channel and a single spare row (or column) of processing elements (PE's) along each boundary of the array. In the presence of faulty PE's the general methodology for reconfiguration involves replacing every faulty PE logically (rather than physically) by a spare PE through a sequence of logical substitutions; these sequences of substitutions are referred to as compensation paths. The determination of such compensation paths for every faulty PE has to be followed by an algorithm to connect each PE to its logical neighbors. It is easy to see that if the number of available routing tracks is fixed, then the compensation paths cannot be arbitrary. Hence, an important question to address is: how many tracks should one provide so as to allow a large enough class of compensation paths, and yet keep the hardware redundancy low. In this paper we show that if there exists a set of compensation paths subject only to the constraints of continuity and nonintersection, then routing channels with three tracks are enough for the reconfiguration of the array. This theoretical result matches the empirical observations presented by several researches showing that three-track routing channels are sufficient for reconfiguring most instances. We refer to the underlying model as a 3-track-1-spare model; this is done to distinguish it from other models that not only use multiple tracks but also multiple spare rows (or columns) along each boundary. We present an efficient algorithm for reconfiguration in our 3-track-1-spare model and evaluate its performance. Our experimental results show that the 3-track-1-spare model has much higher reconfiguration probability than other models that use considerably more spare processors.

Index Terms—Efficient polynomial-time reconfiguration algorithms, fault-tolerant Architecture, multi-track-models, processor arrays, WSI technology.

I. INTRODUCTION

An array of identical processing elements is a well-known architecture in VLSI and WSI technology and proves itself very useful in parallel processing applications. Some of the processing elements in a large array are inevitably going to be faulty, and spare PE's and extra routing hardware are often provided so that a fault-free array can be constructed. Such reconfiguration capability can be used to increase the yield, and to guarantee fault tolerance in applications where failure is not permissible. This paper is concerned with the design and analysis of such configurable fault-tolerant arrays.

The general model to be discussed here is shown in Fig. 1 (see also [9], [13], [3], [17]). It consists of an \( n \times k \) array of identical PE's embedded in a flexible rectangular interconnection architecture. Several rows and columns are chosen to be spares in the sense that they are not used for the regular operation of the array unless there is a failure among the nonspares. The reconfiguration capability is provided by multiple tracks running along the grid lines (the number of tracks present along every grid line is defined as the channel width), and by routing switches that are placed at the intersection of these tracks.

Given a set of faulty PE's, the objective is to reconfigure the connections among the PE's so that a new rectangular logical array is formed comprising only the healthy PE's and demanding no more hardware resources (i.e., spare PE's, tracks, and switches) than available. It can be easily observed that the more the additional hardware, the higher is the reconfiguration probability. However, space and cost limitations might make it impossible to add as much hardware as one would want. Such considerations lead to the following question: What is the optimal amount of required hardware resources,—i.e., the number of spare PE's, the channel width, and the distribution of routing switches,—such that the resulting architecture has high reconfigurability? A related important question addresses the ease of reconfiguration: Given a configurable architecture with fixed resources, are there efficient and simple algorithms for reconfiguring such architectures with high probability?

In this paper, we provide partial answers to both the above issues. However, to further motivate our results and
A. Background

A general methodology to reconfigure arrays with faulty PE's is to determine the so-called compensation paths. A compensation path comprises a sequence of substitutions that logically replaces a faulty PE by a spare one, and can be described as follows. Let a nonspare PE at location \((x, y)\) be faulty, then in any valid reconfiguration it has to be replaced by a healthy processor. Let the faulty PE at location \((x, y)\) be logically replaced by a healthy PE, say at location \((x', y')\); logical replacement implies that in the reconfigured array the physical PE at location \((x', y')\) will be reindexed as \((x, y)\). The PE at location \((x', y')\) is in turn replaced by a healthy PE at location \((x'', y'')\); one can continue this chain until one ends up at a spare PE. Now a compensation path can be defined as the ordered sequence of nodes \((x, y), (x', y'), (x'', y''), \ldots\) involved in the replacement chain. Thus, the compensation paths determine neighbors of each PE in the logical or the reconfigured array. Hence, once the compensation paths are determined, the reconfiguration procedure is completed by connecting each PE to its logical neighbors.

It is easy to see that if the number of faulty PE's is less than the number of nonfaulty spare PE's, then one can always define a set of compensation paths for successful reconfiguration. For example, one could define a set of compensation paths, where each faulty PE is directly replaced by a nonspare faulty PE. Such a direct replacement strategy might require a large number of routing tracks and switches, and the objective is to design more economical compensation paths so that the required set of hardware resources is minimized. In general, the characteristics of the compensation paths (e.g., the geometrical distances between consecutive nodes, or the relative positions of the nodes in the grid) determine the amount of routing hardware needed to implement the necessary connections among the logical neighbors. It can be easily shown that if the number of routing tracks is fixed, then one cannot allow arbitrary sets of compensation paths. In other words, by limiting the hardware resources one limits the number of fault patterns that one can reconfigure. Hence, a natural question to ask is how many tracks should one provide so as to allow a large enough class of compensation paths, and yet keep the hardware redundancy low?

A model with very limited hardware resources has been studied in [7], [8], [14]. It consists of an \(m \times n\) array of nonspare PE's, 1 row (or column) of spare PE's along each boundary, a single-track along every grid line (i.e., channel width = 1), and single-track switches located at the intersections where processors are connected to the grid. It is further assumed that a faulty PE can be converted into a connecting element, thereby making an implicit assumption that there is an extra channel within every PE. The latter assumption has led to the nomenclature of an \(1^{-}\)-track model; however, to emphasize the fact that such a model has only a single row (or column) of spare PE's along every boundary, we shall refer to it as a \(1^{-}\)-track-1-spare model (this is necessary to distinguish it from other models that not only use \(m^{1/2}\) tracks but also use \(m\) spare rows (or columns) along each boundary).

We now briefly discuss the results about the \(1^{-}\)-track-1-spare model. A set of sufficient conditions for determining whether an array with a particular distribution of faulty processors is reconfigurable was derived in [8]. The sufficient conditions that restrict the compensation paths to be straight and continuous can be stated as follows:

An \(m \times n\) array of nonspare PE's, with spare PE's along the sides is reconfigurable into an \(m \times n\) array of healthy processors in a \(1^{1/2}\)-track-1-spare model if 1) there exists a set of continuous and straight compensation paths covering all the faulty nonspare PE's and 2) there is neither intersection nor near-missing among the compensation paths.

A near-miss occurs if two compensating paths in neighboring rows (columns) overlap and run in opposite directions. When a near-miss situation arises, one can show that one track is not sufficient to handle the reconfiguration; hence, near-miss situations are to be avoided.

**Definition 1:** A compensation path is said to be continuous if any two neighboring nodes on a compensation path differ in their coordinates by exactly one, in just one of their coordinates. In other words, a compensation path is continuous if it runs along the grid lines and does not skip any node.

We must note here that the above set of sufficient conditions are not necessary, i.e., a given array can be reconfigured even though a set of compensation paths meeting the above conditions cannot be found (see [12] for examples).

An exponential time algorithm for checking whether a given array can be configured according to the sufficient conditions was first presented in [8]. A polynomial time algorithm \(O(|F|^2\log |F|)\), where \(|F|\) is the number of faulty PE's for the same problem was presented in [12] and was later improved to an \(O(|F|\log |F|)\) time-complexity algorithm in [1].

The algorithms discussed so far focus on the satisfiability question, i.e., whether all the faulty PE's can be replaced by straight and nonintersecting compensation paths. Often, however, a more relevant issue might be to determine the maximum number of faulty processors that can be replaced. In [2], a polynomial time algorithm of time complexity \(O(|F|^{3/2})\) was developed for solving the corresponding combinatorial problem.

The \(1^{-}\)-track-1-spare model thus satisfies the criterion of easy reconfigurability. However, because of the very limited hardware resources in this model, the spare processors cannot be always fully utilized, thus leading to unsatisfactory reconfiguration probability (see Fig. 12 for simulation results on the \(1^{-}\)-track-1-spare model). In fact, one can construct very simple configurations with a constant number of faulty PE's, such that, even though the number of spare PE's grows linearly with the size of the array, one can never reconfigure it in the \(1^{-}\)-track-1-spare model.

Recognizing the above-mentioned limitations, a more hardware intensive model has been proposed in [4], and we shall refer to it as the \(m^{1/2}\)-track-\(m\)-spare model. It consists of an \(n \times k\) rectangular array with \(m\) rows/columns of spare PE's on the four sides. There are \(m\) tracks along every grid line.
of the array and the switches are located at the intersection where PE's are connected to the grid (it is further assumed that the faulty elements can act as connecting elements, thus adding an extra 1/2 track).

A set of sufficient conditions for reconfiguration in the \(m \frac{1}{2}\)-track-\(m\)-sparse model has been presented in [4] by a simple generalization of the conditions in the \(1 \frac{1}{2}\)-track-1-spare model. These sufficient conditions still restrict the compensation paths to be straight; however, because of multiple spare rows and columns, up to \(m\) paths are allowed to overlap along any row or column of the grid. A polynomial time algorithm was presented in [14] for checking whether a given array can be reconfigured according to these generalized sufficient conditions when \(m\) is arbitrary; this answers the open question raised in [4], where an exponential time algorithm was presented for the special case of \(m = 2\).

The major drawback of the \(m \frac{1}{2}\)-track-\(m\) -sparse model (apart from the fact that it requires larger silicon area because of the larger number of tracks) seems to be that it introduces a larger number of spare PE's and yet, at least for small values of \(m\) (e.g., \(m = 2\)), the spare PE utilization is not satisfactory (see Fig. 11 and [4] for simulation results).

We should note here that there are other models of reconfigurable arrays, which do not necessarily use multiple tracks for fault tolerance. For example, TMR (triple modular redundancy) and MMR (multiple modular redundancy) models, which incorporate fault tolerance via a majority voting scheme, have been studied in [10]. A detailed discussion of the various models and a comparison of their relative efficiency and performance can be found in [15].

B. New Results and an Outline of the Paper

In this paper, we address two important issues in this area, namely 1) How to develop architectures that have provably high reconfiguration probability, and 2) How to design efficient reconfiguration algorithms?

We introduce a new model called the 3-track-1-spare model. As shown in Fig. 1, it consists of an \(n \times k\) rectangular array with one row/column of spares in each side. There are three tracks running along every grid line and the switches are located as shown in Fig. 1. Note that we do not require the faulty PE's to be able to act as interconnection elements, as is required in the models discussed earlier.

The motivation behind the definition of the 3-track-1-spare model is to introduce a minimal amount of additional hardware that will allow a much larger class of compensation paths than the restricted class of straight paths. It was shown in [12] that, even in the \(1 \frac{1}{2}\)-track-1-spare model, if one introduces more switches, then isolated bent paths can be allowed, leading to better utilization of the spare processors. The question whether a constant number of tracks can support any set of bent compensation paths was left open. In section II of this paper we prove that if there exists a set of compensation paths subjected only to the constraints of continuity and nonintersection, then the 3-track-1-spare model can always accommodate such a set. Our proof is developed by first constructing a general configuration procedure that works for any set of compensation paths (whether continuous or discontinuous). We then show that three tracks are sufficient to support continuous compensation paths that are reconfigured using this general principle.

These results provide the first known theoretical justification of the observations made by several researchers about the power of 3-track models (see [5], [16]). For example, based on extensive simulations, it is concluded in [5] that, "3 vertical and 3 horizontal data tracks are sufficient to allow reconfiguration." However, the design of a precise model that uses three tracks and is provably powerful does not seem to have been successfully carried out in the literature.

The 3-track-1-spare model also leads itself to easy configuration. For a given array with faulty PE's, a polynomial time algorithm for determining a set of bent compensation paths is presented in Section III (see also [12]). If there is no such set of compensation paths that covers all the faulty PE's, then the algorithm returns a set of paths to cover the maximum number of faulty PE's. The key idea is to reduce the problem to that of a Max-Flow problem in networks, for which several efficient algorithms are known [11].

The simulation results presented in Section III show that the 3-track-1-spare model performs, as was expected, much better than the \(1 \frac{1}{2}\)-track-1-spare model. More remarkably, however, it performs much better than the corresponding \(2 \frac{1}{2}\)-track-2-spare models. Recall that the \(2 \frac{1}{2}\)-track-2-spare model uses twice as many spare PE's and only 1/2 track less than the corresponding 3-track-1-spare model; thus the 3-track-1-spare model does much better with only half the number of spare processors.

II. Reconfiguration in 3-Track-1-Spare Models

In this section, we shall prove that if there exists a set of compensation paths that are continuous and nonintersecting and cover all the faulty PE's, then one can always reconfigure the array using the 3-track-1-spare model. We shall say that a model allows a given set of compensation paths if the routing required for connecting each PE to its logical neighbors (as determined by the set of compensation paths) can be implemented without exceeding the hardware resources defined in the given model. The theorem can be stated as follows (for proof see Section II-D):

\textbf{Theorem 1:} A 3-track-1-spare model allows any given set of continuous and nonintersecting (hence, nonoverlapping) compensation paths.

Recall that the set of compensation paths determines the logical neighbors of each PE. We shall provide a constructive proof for the above theorem, by first developing a general reconfiguration procedure that specifies how to connect each PE to its logical neighbors as specified by the compensation paths. We shall then complete the proof by showing that by following the reconfiguration procedure one does not require more than three tracks.

The idea behind our reconfiguration procedure is very simple and makes only local changes in the interconnection patterns. Let us define the neighbors of PE at \((t, p)\) to be the PE's that \((t, p)\) is supposed to be connected to. For example, PE at \((t + 1, p)\) in Fig. 2 is the bottom neighbor of PE \((t, p)\).
The principles of the reconfiguration procedure can now be stated as follows:

**Reconfiguration Principles:**

1) The connections among the PE's are initially set according to the original grid, meaning that every PE is connected to its top, bottom, right, and left switches. Nothing changes in the connections of a PE if it is not on a compensation path. For example, in Fig. 2, nothing changes in the connection of PE \((i,j)\) if \((i,j)\) is not on a compensation path.

2) Suppose now that PE at \((i,j)\) is on a compensation path \(p\) and replaces PE at \((k,l)\). Then, some of the neighbors of PE at \((i,j)\) are determined by just the compensation path \(p\) itself. For example, in Fig. 2 (where PE \((i,j+1)\) replaces PE \((i,j)\)), the bottom and right neighbors of the PE \((i,j+1)\) in the reconfigured array are PE \((i,j)\) and PE \((i-1,j+1)\), respectively; these are determined by only considering the compensation path that the PE \((i,j+1)\) is on. Such neighbors will be defined as the **predetermined** neighbors.

The neighbors of a PE at \((i,j)\) that cannot be determined by just considering the compensation path PE \((i,j)\) is on are defined as the **undetermined** neighbors. For example, in Fig. 2, the left and top neighbors of PE at \((i,j+1)\) are not determined if we only consider the path PE \((i,j+1)\) is on; hence, they are the **undetermined** neighbors of PE \((i,j+1)\). Note that if PE's at \((i-1,j)\) and \((i,j-1)\) are not on a compensation path, then they are PE \((i,j+1)\)'s top and left logical neighbors, respectively. On the other hand, if they are on some compensation path, the PE's that replace them are the PE \((i,j+1)\)'s top and left logical neighbors.

Following the above definitions, the connections of a PE \((i,j)\), on a compensation path \(p\) are set as follows:

- **a)** If right (left, top, bottom) neighbor of PE \((i,j)\) is a predetermined one, then connect the right (left, top, bottom) switch of PE \((i,j)\) with the left (right, bottom, top) switch of its right (left, top, bottom) neighbor. For example, the predetermined right neighbor of PE \((i,j+1)\) in Fig. 2 is PE \((i-1,j+1)\); so we connect the right switch of PE \((i,j+1)\) to the left switch of \((i-1,j+1)\).

- **b)** If right (left, top, bottom) neighbor of PE \((i,j)\) is undetermined, then connect the right (left, top, bottom) switch of PE \((i,j)\) with the right (left, top, bottom) switch of PE \((k,l)\) that \((i,j)\) is replacing. For example, the left neighbor of PE \((i,j+1)\) in Fig. 2 is an undetermined one. So we connect the left switch of PE \((i,j+1)\) with the left switch of PE \((i,j)\) that \((i,j+1)\) is replacing.

3) An optimization of the number of tracks used is applied when two tracks that run in parallel get connected in a switch later on. In such a case the two tracks get connected in the first switch they meet before they run parallel to each other. Fig. 4(b) compared to Fig. 4(c) illustrates this optimization principle.

**Lemma 1:** The reconfiguration of an array according to the Reconfiguration Principles guarantees that the connections among the PE's are set in a correct way in the reconfigured array.

**Proof:** We shall prove that the Reconfiguration Principles guarantee that if two PE's are neighbors in the reconfigured array (as determined by the compensation paths), then the corresponding links of the PE's are always connected to a common switch (hence, the link can be set up by programming the switches appropriately). Recall that a PE lying on a compensation path has two types of neighbors: **predetermined** and **undetermined.** As far as the predetermined neighbors are concerned, the second principle guarantees that every PE \((i,j)\) gets connected to the same switches with its predetermined neighbors. For example, if the right predetermined neighbor of PE \((i,j)\) is PE \((k,l)\), then the right switch of PE \((i,j)\) gets connected to the left switch of PE \((k,l)\) and the connection is completed.

Let us now discuss the undetermined neighbors. Let us suppose that PE \((i,j)\) replaces PE \((k,l)\), and that the right neighbor of PE \((i,j)\) is undetermined. If the physical right neighbor of \((k,l)\), (i.e., \((k,l+1)\)), is not on a compensation path, then the logical right neighbor of PE \((i,j)\) in the reconfigured array should be \((k,l+1)\). If \((k,l+1)\) is on
VARVARIOU et al.: RECONFIGURING PROCESSOR ARRAYS USING MULTIPLE-TRACK MODELS

A compensation path and is replaced by PE \((f, g)\), then \((f, g)\) is the logical right neighbor of PE \((i, j)\) in the reconfigured array. For the sake of generality let’s assume that the logical right neighbor of PE \((i, j)\) is PE \((f, g)\); this also implies that the left neighbor of PE \((f, g)\) is undetermined. Now, according to the Reconfiguration Principles, the following connection are made: 1) Since the right neighbor of PE \((i, j)\) is undetermined, the right switch of PE \((i, j)\) is connected to the right switch of the PE \((k, l)\). 2) Since the left neighbor of PE \((f, g)\) is undetermined, and since PE \((f, g)\) is replacing PE \((k, l + 1)\), the left switch of PE \((f, g)\) is connected to the left switch of PE \((k, l + 1)\). However, the right switch of PE \((k, l)\) is the same as the left switch of PE \((k, l + 1)\). Hence, the right link of PE \((i, j)\) is connected to the same switch as the left link of its logical neighbor \((f, g)\).

The third Reconfiguration Principle does not disturb the physical connection that have been achieved so far. The objective is to reduce redundancy by eliminating wires that need to be connected but are running parallel along certain segments of the tracks.

Now, the only thing we have to show is that the above reconfiguration techniques do not require more than three tracks. The presentation, however, is greatly facilitated if reconfiguration of various restricted types of compensation paths is illustrated first. The proof that three tracks are sufficient will follow easily from those for the special cases. In particular, we shall study compensation paths that are restricted to be:

**Case 1:** only straight without any near-miss situations.

**Case 2:** only straight with possible near-miss situations.

**Case 3:** only completely bent (or staircase).

**Remark:**

- There are eight switches and eight channel-segments (each channel-segment has three tracks) that surround a PE, as illustrated in Fig. 3. For the sake of consistency, the switches and the channel-segments around a PE at \((i, j)\) will be numbered as shown in Fig. 3. For example:

  1. Switch \(-t_{i,j}\) is located on top of the PE at \((i, j)\), and Switch \(-r_{i,j}\) is located on top-right of the PE at \((i, j)\);

  2. Channel \(-t_{i,j}\) is located on top-left of the PE \((i, j)\).

**A. Reconfiguration for Case 1**

When the compensation paths are straight and without nearmisses, then a single-track model is sufficient and the general reconfiguration procedure is already presented in [8]. We, however, briefly review it here for the sake of completeness and also to introduce certain basic and useful concepts that are necessary for the other cases.

**Lemma 2:** The 3-track-1-spare model can support reconfiguration of a set of compensation paths of Case 1.

**Proof:** Consider first a compensation path that does not run in parallel with any other compensation path, as the one shown in Fig. 4(a). Without loss of generality, assume that the path starts at processing element \((i, j)\) and runs downward vertically. Do the following changes in the already existing connections (the changes are illustrated in Fig. 4(a)):

1. Bypass the faulty PE doing the following: Connect Switch \(-t_{i,j}\) with Switch \(-t_{i+1,j}\) occupying a track in Channel \(-tr_{i,j}\), Channel \(-r_{i,j}\), Channel \(-rb_{i,j}\) and Channel \(-br_{i,j}\).

2. For each processing element \((k, l)\) on the compensation path, other than the faulty one, PE’s at \((k-1, l)\) and \((k+1, l)\) are its predetermined top and bottom neighbors. The necessary connections are already set, so no changes are required.

3. For each processing element \((k, l)\) on the compensation path (that replaces \((k-1, l)\)), its left and right neighbors are undetermined so, according to the second principle, so the following: a) Connect Switch \(-r_{k,l}\) with Switch \(-r_{k-1,l}\) occupying a track in Channel \(-r_{k,l}\) and Channel \(-lb_{k-1,l+1}\). b) Connect Switch \(-l_{k,l}\) with Switch \(-l_{k-1,l}\) occupying a track in Channel \(-lt_{k,l}\) and Channel \(-rb_{k-1,l-1}\).
Now consider the case when more than one compensation path runs downward in adjacent columns. Then make the following changes in the already existing connections:

1) Apply the procedure presented above for each compensation path.
2) Apply the optimization principle to reduce redundancy by eliminating wires that need to be connected, but run in parallel along certain segments of the tracks.
Figs. 4(b) and (c) illustrate such a reconfiguration procedure.

B. Reconfiguration for Case 2

Recall here that the near-miss situation arises when two compensation paths running in adjacent rows/columns in opposite directions overlap (see Fig. 5(a)).

We shall define the size of a near-miss situation as follows:

Definition 2: The size of a near-miss between two compensation paths is the number of PE’s in each compensation path that overlap with the PE’s in the other compensation path.

For example, the near-miss of Fig. 5(a) is of size 4.

We shall now prove that any near-miss can be reduced in size, using bent paths, to a near-miss of size at most 2. Hence, it is sufficient to show how to reconfigure compensation paths with near-miss of size 2 and then show how to accommodate bent paths.

Proposition 1: Given any set of compensation paths with near-miss situations of any size, we can find another valid set of compensation paths where there is no near-miss of size greater than 2.

Proof: The constructive proof follows directly from Fig. 5(b): If the paths overlap by more than two nodes, then introduce bends as shown in the figure and remove the near-miss situation. Note that if the near-miss is of size 2, it cannot
be eliminated by just introducing bent paths; if, for example, one eliminates a near-miss situation of size 2 in the horizontal direction, one creates a near-miss of size 2 in the vertical direction.

So from now on we can assume that the largest size of a near-miss situation that can appear is 2. If the near-miss is of size greater than 2 then we reduce its size before proceeding to any other reconfiguration step.

**Lemma 3:** The 3-track-1-spare model can support reconfiguration of a set of compensation paths of Case 2.

**Proof:** As stated in Proposition 1, the largest size of near-miss that can appear in a set of compensation paths is 2.

Consider first a case where two straight compensation paths run, without loss of generality, vertically, in opposite directions, the first one starting at PE \((i, j)\) going downward and the second one starting at \((i + l, j + 1)\) going upward as shown in Fig. 5(c), creating a near-miss situation of the largest possible size 2. Then applying the reconfiguration procedure for the simple straight paths presented in Lemma 1, we reconfigure the near-miss case as well. The reconfiguration for more than one near-miss in adjacent columns/rows is exactly the same as shown in Fig. 5(c).

### C. Reconfiguration for Case 3

A bent compensation path will be defined as completely bent or as a staircase if it has no straight segment of length more than 1, i.e., it has successive right-angle bends, as illustrated in Fig. 6(a). We should note here that the reconfiguration for such staircase paths is, by far, the most difficult to achieve. Moreover, once such completely bent paths are reconfigured, other bent paths can be easily handled by breaking them into straight and completely bent segments (more details are given later).

Before we proceed, however, let us define a near-miss situation among bent compensation paths: Two bent compensation paths generate a near-miss situation only if they run next to each other in opposite directions. The following proposition proves an important fact: Near-miss situations among bent paths can always be removed.

**Proposition 2:** Near-miss situations among bent paths can always be removed.

**Proof:** Without loss of generality, consider two compensation paths \(p_1\) and \(p_2\) with near-miss, as shown in Fig. 7(a). The paths can be broken up into segments as follows:

\[
p_1 = x_1((i, j), z_1, (k, l))y_1
\]

and

\[
p_2 = x_2((k, l + 1), z_2, (i + l, j))y_2
\]

Segments \(x_1, x_2\) and \(y_1, y_2\) are without near-misses, whereas segments \(((i, j), z_1, (k, l))\) and \(((i, j), z_2, (k, l))\) generate a near-miss between them. (In the general case, \(p_2 = x_2((m, n), z_2, (p, s))y_2\) columns/rows is exactly the same as shown in Fig. 4(c), where \((i, j)\) and \((k, l)\) are close neighbors of \((m, n)\) and \((p, s)\) respectively.)

Now, define a new set of compensation paths replacing compensation paths \(p_1\) and \(p_2\) by the \(p'_1\) and \(p'_2\) constructed as follows (see Fig. 7(b)):

\[
p'_1 = x_1((i, j), (i + 1, j))y_1
\]

and

\[
p'_2 = x_2((k, l + 1), (k, l))y_2
\]

Note that the near-miss in bent paths is of size greater than 2, so the new compensation paths \(p'_1\) and \(p'_2\) do not near-miss at all.

The above proposition implies that for completely bent paths one need not consider near-miss situations.

**Lemma 4:** The 3-track-1-spare model can support reconfiguration of a set of compensation paths of Case 3.

**Proof:** We shall first show how to reconfigure around a single staircase path using at most three tracks; we shall then show that if many such paths run next to each other, then also the restriction of three tracks is sufficient.

Consider the completely bent path of Fig. 2. Let us first show how to reconfigure horizontal substitution, e.g., substi-
tution of PE at (i, j) by PE at (i, j + 1). The neighbors of PE at (i, j + 1) in the configured array are determined as follows:

- According to the compensation path, PE (i, j) replaces PE at (i + 1, j); hence, PE at (i, j) becomes the bottom predetermined neighbor of PE (i, j + 1).
- PE at (i − 1, j + 1) replaces PE at (i, j + 1); hence, PE (i − 1, j + 1) becomes the right predetermined neighbor of PE (i, j + 1).
- The top and left neighbors of PE at (i, j + 1) on this substitution are undetermined. In general, the top and left neighbors of PE (i, j + 1) are going to depend on the kind of compensation paths PE (i − 1, j) and PE (i, j − 1) are on. So according to the second reconfiguration principle, connect top switch of PE (i, j + 1) to the top switch of PE (i, j) (or bottom switch of PE (i, j + 1)) and left switch of PE (i, j + 1) to the left switch of PE (i, j) (or right switch of PE (i, j − 1)).

The above idea can be considered as the centerpiece that allows our reconfiguration algorithms to work, and it is worthwhile to reiterate the justification. Consider, for example, the PE at (i − 1, j):

Case 1: PE (i − 1, j) is not on any compensation path so its bottom neighbor is PE (i, j + 1). The connection required is trivial: We have already connected the top link of PE (i, j + 1) to the bottom switch of PE (i − 1, j).

Case 2: PE (i − 1, j) is replaced by some other PE (say PE (k, l)), so the bottom neighbor of the PE (k, l) is PE (i + 1, j + 1). However, by our reconfiguration strategy, since PE (k, l) replaces PE (i − 1, j), its bottom logical neighbor is undetermined and hence the bottom link of PE (k, l) is connected to the bottom switch of PE (i − 1, j). Hence, PE (i, j + 1) can be the bottom neighbor of PE (k, l) by just connecting the corresponding links via the bottom switch of PE (i − 1, j).

Fig. 7 illustrates the horizontal replacement scheme discussed above. The dashed line connects PE (i, j + 1) with its new bottom neighbor PE (i, j). The dotted line connects (i, j + 1) with its new right neighbor (i − 1, j + 1). The solid line connects the left link of (i, j + 1) with the left switch of (i, j) as well as the top link of PE (i, j + 1) with the top switch of PE (i, j).

The connections required and the resulting occupation of the tracks for the horizontal substitution of PE (i, j) by PE (i, j + 1) can be stated in a code form as follows.

1) Connect Switch − t_{i,j+1} to Switch − t_{i,j} using Channel − t_{l_{i,j+1}} and Channel − t_{r_{i,j}}.
2) Connect Switch − r_{i,j+1} to Switch − l_{i−1,j+1} using Channel − r_{l_{i,j+1}} and Channel − r_{t_{i,j+1}} and Channel − l_{b_{i−1,j+1}}.
3) Connect switch − b_{i,j+1} to Switch − t_{i,j} using Channel − b_{l_{i,j+1}} and Channel − t_{l_{i,j+1}} and Channel − r_{t_{i,j}}.
4) Connect Switch − l_{i,j+1} to Switch − l_{i,j} using Channel − b_{l_{i,j+1}} and Channel − r_{t_{i,j}} and Channel − b_{l_{i,j}}.

Similarly, for the vertical substitution of PE (i + 1, j) by PE (i, j) (see Fig. 6(a)):

- PE (i+1, j) replaces PE (i+1, j−1); hence, PE (i+1, j) will be the predetermined left neighbor of (i, j).
- PE (i, j + 1) replaces PE (i, j); hence, PE (i, j + 1) becomes the top predetermined neighbor of PE (i, j).
- The bottom and right neighbors of PE (i, j) are undetermined, so according to the second principle, we connect the bottom and right switches of (i, j) to the bottom and right switches of (i + 1, j) PE, which (i, j) is replacing.

Fig. 6(a) illustrates a vertical substitution. The dashed line connects (i, j) PE to its new top neighbor (i, j + 1). The dotted line connects (i, j) PE to its new left neighbor (i + 1, j). Solid lines connect the bottom and right switches of (i, j) PE with the bottom and right switches of (i + 1, j) PE.
Fig. 8. Reconfiguration for special case 3: The track occupancy of each channel is shown to be at most 3.

4) Connect Switch - \( i_{1,j} \) to Switch - \( b_{1,j+1} \) using Channel - \( r_{1,j} \), Channel - \( r_{i,j} \), Channel - \( r_{b_{1,j}} \), and Channel - \( b_{1,j+1} \).

Fig. 6(b) illustrates the superimposition of the horizontal and vertical substitutions along the bent path. Fig. 8(a) shows the resulting occupancy of the channels. The numbers shown are the numbers of communication tracks occupied in each segment of the routing channels. \textit{Note that none of these numbers exceeds 3.}

Consider now the two parallel bent paths of Fig. 8(b) that run next to each other: Each can be reconfigured according to the principles described above. The number of necessary tracks along each channel-segment will be the sum of the tracks used by the leftmost bent path, whereas the numbers on the right correspond to tracks occupied by the rightmost bent path. \textit{Note that the total number of tracks occupied by both bent paths never exceeds 3. Therefore, the 3-track-1-spare model is capable of handling the reconfiguration for any set of paths corresponding to Case 3.}

\section*{D. Reconfiguration of the General Case}

In this section we are going to describe how to handle the general case where the compensation paths can be arbitrary. The procedure of assigning tracks to the necessary connections will be based on the way the special cases were treated. The proof that the 3-track-1-spare model can accommodate any set of continuous and nonintersecting paths will also follow from those of the special cases. We shall define mixed compensation paths as follows:

\textit{Definition 3:} A mixed compensation path is a compensation path that has both straight and bent parts.

\textit{Note that mixed compensation paths are the most general types of paths one can have.}

\textit{Theorem 1:} A 3-track-1-spare model allows any given set of continuous and nonintersecting (hence, nonoverlapping) compensation paths.

\textit{Proof:} Since we have already discussed straight paths and completely bent paths, in order to prove the theorem, we have to show that three tracks are sufficient for any set of mixed compensation paths. We shall show this by considering 1) isolated mixed compensation paths, 2) mixed compensation paths that run in parallel, and 3) mixed compensation paths with a near-miss situation.

First consider the isolated mixed compensation path of Fig. 9(a): PE \((i,j)\) is in the boundary of the bent part \(p_1\) and the straight part \(p_2\) of the compensation path \(p = p(i,j)p_2\). So far we know how to deal with \(p_1\) and \((i,j)p_2\) separately, according to cases 3 and 1, respectively. Applying the steps of the corresponding reconfiguration procedures for \(p_1\) and \((i,j)p_2\) we can find a valid reconfiguration for the case of the mixed path \(p\) as well. Fig. 9(b) illustrates the necessary connections and the corresponding occupancy of the tracks for the case of the mixed path \(p\). The proof that this is a valid reconfiguration using three tracks is immediately apparent from the proofs for Cases 1 and 3.

Now, if two mixed paths run in parallel next to each other, then from simple geometrical considerations, it is obvious that either they have completely bent sections running in parallel or straight sections running in parallel. For both special cases we showed how to reconfigure with only three tracks. Fig. 9(c) illustrates an example of two parallel mixed paths and the corresponding occupancy of the tracks.

The only case left is when two paths run opposite, but next, to each other; we have referred to such situations as near-miss situations. However, from Propositions 1 and 2, we know that such a situation can occur only for a straight segment of length 2. The procedure to handle such a situation with three tracks was discussed in Special Case 2.

Having explored all the possible patterns of compensation paths, we conclude that the 3-track-1-spare model is capable...
of handling the reconfiguration for any general pattern of the compensation paths.

III. COMPLEXITY AND EVALUATION OF THE 3-TRACK-1-SPARSE MODEL

So far we have shown that if there exists a set of continuous and nonintersecting compensation paths that cover all the faulty PE's, then the 3-track-1-spare model can always handle the reconfiguration. In the first part of this section we briefly describe an algorithm for determining whether such a set of compensation paths exists. If it exists, then the algorithm also returns one possible set; if such a set does not exist then, the algorithm returns a set that covers the maximum possible number of faulty PE's. In the second part of this section we discuss the performance of the 3-track-1-spare model and compare it to the performance of other models that use a similar type and amount of additional hardware.

A. Determining a Set of Nonintersecting Continuous Compensation Paths

The problem of determining a set of nonintersecting continuous compensation paths in a faulty array can be formally stated as follows:

Problem 1:

Let $V$ be the set of grid points in an $n \times n$ two-dimensional rectangular grid, and let $F \subset V$. Determine a set of nonintersecting paths in the grid such that: 1) Each vertex $v \in F$ is connected on the boundary of the grid by one of the paths and 2) the paths are nonintersecting.

The problem above reduces to a Max-Flow problem as follows (see Fig. 10).
1) Define $V' = V \cup \{s, t\}$, where $V$ is the set of the nodes in the array, $s$ is the source node, and $t$ is the sink node.
2) Define the set of arcs $E'$ between the nodes of $V'$ as follows: 1) for every pair of nodes $(i, j)$ that are adjacent in the grid, define two arcs $i \rightarrow j$ and $j \rightarrow i$; 2) for every boundary node $v \in V$, define $v \rightarrow t$; and 3) for every node $v \in F$, define an arc $s \rightarrow v$ (see Fig. 10).
3) Define the capacity of every edge to be unity.
4) Define the capacity of every node to be unity.
5) Solve the Max-Flow problem for the graph constructed above. Each compensation path in our original problem is now defined by a unit flow from $s$ to $t$ in the network.

The complexity of the Max-Flow algorithm for general networks is $O(n^3)$ when the capacity of the nodes is unity.

Disregarding source node $s$ and the edges that are adjacent to it, and considering all nodes $v \in F$ to be the new source of the network, the grid becomes a planar multisource single-sink network. The demand of every source node $v \in F$ is now fixed to be $d(v) = 1$, meaning that a unit flow is required from every source node and corresponds to a compensation path for every faulty node of our original grid. The demand of
the sink $t$ is now fixed to be $d(t) = |F|$. So the flow network of Fig. 10 becomes a planar multisource single-sink graph, with unity capacities in the edges and the nodes, and fixed source and sink demands. The flow problem in such graphs can be reduced to a circulation problem (with lower bounds on the edge capacities) as shown in [6], and can be solved in $O(n^{1.5} \log n)$ time.

B. Performance of the 3-Track-1-Spare Model

The simulation results presented here were performed by injecting random faults in an array, and then determining whether a set of continuous and nonintersecting compensation paths can be found to cover all the faulty PE's; the algorithm described above was used for the latter part. We simulated arrays of various sizes ($4 \times 4, \ldots, 25 \times 25$) and we randomly injected faults into these arrays. We have computed the Array Yield (defined as: # of Reconfigured Arrays/Total # of Simulated Arrays) in terms of the PE Yield (defined as: # of Healthy PEs/Total # of PEs) as shown in [6].

IV. CONCLUDING REMARKS

In this paper, we studied a 3-track-1-spare model that has three tracks along every channel and one spare row or column of spare PEs. This model is better than the $2\frac{1}{2}$-track-2-spare model, which has similar additional hardware. The Array Yield for our model is in several cases up to 50% better than the Array Yield for the $2\frac{1}{2}$-track-2-spare model for the same PE Yield. The diagrams of Fig. 12 illustrate the simulation results of $1\frac{1}{2}$-track-1-spare model that has the same number of PEs. It is obvious that our model performs much better than the $1\frac{1}{2}$-track-1-spare model.

Extensive simulations of our model showed that the use of spare PEs is much better in the "interior" of the rows/columns of the spare PE's compared to the use of the spare PE's that are located in the "corners" of the array. This is easy to interpret. Consider the subarray $A$ in the top left corner of the original array, as shown in Fig. 13. Simulations showed that in most of the cases the faulty PE's in $A$ get reconfigured through compensation paths that connect the faulty PE's with the spare PE's that are in part 1 of the spare rows/columns as shown in Fig. 13. So, if we assume a uniform distribution of the faulty PE's over the whole array, the ratio # of spares/ # of faulty PE's is proportional to the ratio # of spares/ # of nonspares $= 2s/s^2$ where $s$ is the size of subarray $A$, which grows as $s$ gets smaller. That means that there is not a very good use of the spare PE's near the "corners" of the array (where $s$ is small) compared to the use of the spare PE's in the "interior" part of the spare rows/columns (where $s$ is large). So a possible way of increasing the Array Yield in terms of the PE Yield would be to increase the number of spare PE's in the "interior" part of the spare rows/columns and decrease it in the "corners" of the array.
column along each boundary. We showed that it uses the spare but continuous compensation paths. This provides theoretical justification of the observations made in the literature (see e.g., IEEE TRANSACTIONS ON COMPUTERS, vol. 39, no. 1, Jan. 1990, pp. 234-250).

Following are some related remarks and open issues.

1) The reconfiguration principles introduced in Section II can be applied for arbitrary kinds of compensation paths (i.e., overlapping, diagonal, and so on). The use of more general compensation paths may, however, require more than three tracks.

2) Introducing one more track per channel (i.e., a 4-track model) enlarges the class of admissible compensation paths, e.g., one can allow compensation paths that run diagonally or even intersect. One could increase the reconfiguration probability even more by increasing the channel width. However, the question is at what point do the penalties incurred because of additional hardware offset the benefit of higher reconfigurability. The 3-track-1-spare model with high reconfigurability, as well as a small enough number of tracks, seems to be an attractive compromise.

3) The Reconfigurability Principles of this paper can be applied to other topologies such as hexagonal and triangular grids. We can show that equivalent 3-track-1-spare models in such topologies are sufficient to allow any set of nonintersecting compensation paths. Moreover, the algorithm proposed in Section III-A can be used to determine compensation paths in such grids.

4) An important open question is as follows: Given an $n \times n$ array, if the PE's fail independently with probability $p(n)$, then what is the probability that it can be reconfigured by the 3-track-1-spare model? Simulation results show that as long as the faulty PEs are less in number than the spare PE's, then the probability is quite high. However, given the simplicity of reconfiguration in the 3-track-1-spare model, one hopes that a precise analytical solution to the above problem would be possible.

REFERENCES

[12] V. P. Roychowdhury, J. Bruck, and T. Kailith, "Efficient algorithms for reconfiguration in the 3-track-1-spare model and showed that it has much higher reconfiguration probability than the corresponding 2 1/2-track-2-spare model.

Fig. 13. Improving the performance of the 3-track-1-spare model.

Theodora A. Varvarigou was born in Athens, Greece, in 1966. She received the B. Tech. degree from the National Technical University of Athens, Athens, Greece in 1988, the M.S. degree in 1989, and the Ph.D. degree in 1991 from Stanford University, Stanford, CA.

She is currently working at AT&T Bell Labs, Holmdel, New Jersey. Her research interests include distributed telecommunication systems, multimedia and multiparty communications, parallel algorithms and architectures, fault-tolerant computation, and parallel scheduling on multiprocessor systems.
Vwani P. Roychowdhury received the B. Tech degree from the Indian Institute of Technology, Kanpur, India and the Ph.D. degree from Stanford University, Stanford, Calif. in 1982 and 1989 respectively, all in electrical engineering.

From September 1989 to August 1991, he was a Research Associate in the Department of Electrical Engineering at Stanford University. He is currently an Assistant Professor in the Electrical Engineering Department at Purdue University. His research interests include parallel algorithms and architectures, design and analysis of neural networks, special purpose computing arrays, VLSI design, and fault-tolerant computation.

Thomas Kailath (S'57-M'62-F'70) was educated in Poona, India, and at the Massachusetts Institute of Technology (S.M., 1959; Sc.D., 1961).

From October 1961 to December 1962, he worked at the Jet Propulsion Laboratories, Pasadena, CA, where he also taught part-time at the California Institute of Technology. He then came to Stanford University, where he served as Director of the Information Systems Laboratory from 1971 through 1980, as Associate Department Chairman from 1981 to 1987, and currently holds the Hitachi America Professorship in Engineering. He has held short-term appointments at several institutions around the world. He has worked in a number of areas, including information theory, communications, computation, control, signal processing, VLSI design, statistics, linear algebra, and operator theory; his recent interests include applications of signal processing, computation, and control to problems in semiconductor manufacturing and wireless communications. He is the author of *Linear Systems*, (Prentice-Hall, 1980), and *Lectures on Wiener and Kalman Filtering*, (Springer-Verlag, 1981). Dr. Kailath has held Guggenheim, Churchill, and Royal Society fellowships, among others, and received awards from the IEEE Information Theory Society and the American Control Council, in addition to the Technical Achievement and Society Awards of the IEEE Signal Processing Society. He served as President of the IEEE Information Theory Society in 1975, and has been awarded honorary doctorates by Linköping University, Sweden, and by Strathclyde University, Scotland. He is a Fellow of the Institute of Mathematical Statistics and is a member of the National Academy of Engineering.