Reconfiguring Processor Arrays
Using Multiple-Track Models

Theodora A. Varvarigou Vwani P. Roychowdhury Thomas Kailath*

Abstract

In this paper 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 (PEs) along each boundary of the array. In the presence of faulty PEs, 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 subjected 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 researchers showing that 3-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 facilitate 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 it has much higher reconfiguration probability than other models that use considerably more spare processors.

1 Introduction

An array of identical processing elements is a well known architecture in the VLSI and WSI technology and proves itself very useful in parallel processing applications. Often, however, during the fabrication process or during run-time, some of the processing elements in a large array are inevitably going to be faulty. Spare PEs 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 when 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 [5]). It consists of an \( n \times k \) array of identical PEs 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 non-spares. The reconfiguration capability is provided by multiple tracks running along the grid lines (the number of tracks present along every grid line is

*The authors are with the Information Systems Laboratory, Stanford University, CA, 94305, and their work was supported in part by the SDIO/IST U. S. Army Research Office through Contract DAAL03-87-K-0033.
defined as the channel width), and by routing switches that are placed at the intersections of these tracks.

Given a set of faulty PEs, the objective is to reconfigure the connections among the PEs such that a new rectangular logical array is formed comprising only the healthy PEs and demanding no more hardware resources (e.g., spare PEs, tracks, and switches) than available. It is obvious that the more the additional hardware, the higher is the reconfiguration probability. Nevertheless, space and cost limitations might make it impossible to add as much hardware as one would want. Such considerations lead to the following design question: What is the optimal amount of required hardware resources, i.e., the number of spare PEs, 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 contributions, and to introduce some of the relevant concepts we shall first briefly review the previous work in this area.

1.1 Background

A general methodology to reconfigure arrays with faulty PEs is to determine the so-called compensation paths. A compensation path is comprised of a sequence of substitutions that logically replaces a faulty PE by a spare one, and can be described as follows. Let a non-spare 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 \((x, y)\) 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 \((x', y')\) is in turn replaced by a healthy PE, say 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 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 PEs is less than the number of spare PEs, then one can always define a set of compensation paths for successful reconfiguration. However, the characteristics of the compensation paths (e.g., the geometrical distances between consecutive nodes, 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 faulty 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 [3, 4, 6]. It consists of an \(m \times n\) array of non-spare PEs, 1 row (or column) of spare PEs along each boundary, a single-track along every grid line (i.e., channel width = 1), and single-track switches located at 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 each PE. The latter assumption has led to the nomenclature of an \(1\frac{1}{2}\)-track model; however, to emphasize the fact that such a model has only a single row (or column) of spare PEs along every boundary, we shall refer to it as a \(1\frac{1}{2}\)-track-1-spare model (this is necessary to distinguish it from other models that not only use \(m\frac{1}{2}\) tracks but also use \(m\) spare rows (or columns) along each boundary).
We now briefly discuss the results about the 13-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 [4]. The sufficient conditions restrict the compensation paths to be straight and continuous and can be stated as follows:

Given an m x n array of non-spare PEs, with spare PEs along the sides, it is reconfigurable into an m x n array of healthy processors in a 13-track-1-spare model if 1) there exists a set of continuous and straight compensation paths covering all the faulty non-spare PEs and 2) there is neither intersection or near-miss among the compensation paths.

A near-miss situation occurs if two compensation paths in neighboring rows (columns) overlap and are 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.

Also, a compensation path is said to be 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 [6] for examples).

An exponential time algorithm for checking whether a given array can be reconfigured according to the sufficient conditions was first presented in [4]. A polynomial time algorithm $O(|F|^2)$, where $|F|$ is the number of faulty PEs, for the same problem was presented in [6] and was later improved to an $O(|F| \log |F|)$ time-complexity algorithm.

The 13-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. 2 for simulation results on the 13-track-1-spare model). In fact one can construct very simple configurations with a constant number of faulty PEs such that, even though the number of spare PEs grows linearly with the size of the array, one can never reconfigure it in the 13-track-1-spare model.

Recognizing the above mentioned limitations, a more hardware intensive model has been proposed in [1], and we shall refer to it as the m1-track-m-spare model. It consists of an m x k rectangular array with m rows/columns of spare PEs on the four sides. There are m tracks along every grid line of the array and the switches are located at the intersections where PEs 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 m1-track-m-spare model has been presented in [1] by a simple generalization of the conditions in the 13-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 [7] 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 [1], where an exponential time algorithm was presented for the special case of m = 2.

The major drawback of the m1-track-m-spare 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 PEs and yet at least for small values of m (e.g., m = 2) the spare PE utilization is not satisfactory (see Fig. 2 and [1] for simulation results).

1.2 New Results

In this paper we address two important issues in this area, namely (1) How to design architectures that have provably high reconfiguration probability, and (2) How to design efficient reconfiguration
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 PEs 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 [6] that even in the 1-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 as an open question. We can prove the following theorem about the 3-track-1-spare model:

**Theorem 1** 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 reconfiguration procedure that works for any set of compensation paths (whether continuous or discontinuous). We then show that three tracks are sufficient to support any set of continuous compensation paths when reconfigured using this general principle. Because of space limitations the proof is omitted here; a detailed proof can be found in [8].

We must note here that the above result provides the first known theoretical justification of the observations made by several researchers about the power of 3-track models. For example in [2], based on extensive simulations, it is concluded 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 lends itself to easy reconfiguration. For a given array with faulty PEs, a polynomial time algorithm for determining a set of bent compensation paths exists (see [8], [6]); if there is no such set of compensation paths that covers all the faulty PEs, then the algorithm returns a set of paths to cover the maximum number of faulty PEs. The key idea is to reduce the problem to that of a Max-Flow problem in networks, for which several efficient algorithms are known. The simulation results presented in the following section show that the 3-track-1-spare model performs, as was expected, much better than the 1-track-1-spare model. More remarkably, however, it performs much better than the corresponding 2½-track-2-spare models. Recall that the 2½-track-2-spare models use twice as many spare PEs 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.

2 Complexity and evaluation of the 3-track-1-spare model

So far we have shown that if there exists a set of continuous and non-intersecting compensation paths that covers all the faulty PEs, then the 3-track-1-spare model can always handle the reconfiguration. In the first part of this section we shall 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 PEs. In the second part of this section we will discuss the performance of the 3-track-1-spare model and compare it to the performance of other models that use similar type and amount of additional hardware.
2.1 Determining a set of non intersecting continuous compensation paths.

The problem of determining a set of non intersecting 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 $n \times n$-dimensional rectangular grid, and let $F \subseteq V$. Determine a set of non intersecting 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 non intersecting.

The problem above reduces to a MAX-FLOW problem as follows:

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 an arc $v \rightarrow t$, (3) for every node $v \in F$, define an arc $s \rightarrow v$.
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 sources of the network, the grid becomes a planar multi-source 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 becomes a planar multi-source 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) and can be solved in $O(n^{1.5} \log n)$ time.

2.2 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 PEs; the algorithm described above was used for the latter part. We simulated arrays of various sizes ($4 \times 4$, ..., $25 \times 25$) 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) $= \frac{N^2 + 4N - F}{N^2 + 4N}$ where $N$ is the size of the array simulated, $F$ is the number of faults injected in the array, $N^2$ is the number of the non spare PEs, $4N$ is the number of the spare PEs, $N^2 + 4N$ is the total number of the PEs.\)

The results of the simulations are shown in the diagrams of Fig. 2; the corresponding results for the $2\frac{1}{2}$-track-2-spare model as reported in [1] are also shown in Fig. 2. As we have already mentioned this model requires $2$ and $1/2$ tracks and has two rows/columns of spare PEs in each side, whereas our model requires $3$ tracks and only one row/column of spare PEs in every side. One can easily see by comparing the two diagrams that our model does much 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\textsuperscript{1/2}-track-2-spare model for the same PE Yield. The diagrams of Fig. 2 illustrate the simulation results for the 1\textsuperscript{1/2}-track-1-spare model that has the same number of spare PEs. It is obvious that our model performs much better than the 1\textsuperscript{1/2}-track-1-spare model.

3 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 along each boundary. We showed that it uses the spare processors very efficiently; more precisely, we proved that a 3-track-1-spare model can support any set of non-intersecting compensation paths. This provides theoretical justification of the observations made in the literature (see e.g., [2]) about the power of 3-track models. We also discussed efficient algorithms for reconfiguration in the 3-track-1-spare model and showed that it has much higher reconfiguration probability than the corresponding 2\textsuperscript{1/2}-track-2-spare model. Note that the reconfiguration principles introduced in our paper can be applied for arbitrary kinds of compensation paths (i.e., overlapping, diagonal etc.). The use of more general compensation paths may, however, require more than three tracks. Moreover, the reconfiguration 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 non-intersecting compensation paths.

References


Figure 1: The 3-track-1-spare Reconfiguration Model.

Figure 2: Simulation results.