On Reconfiguration Methods of Processor Arrays Based on Sequential Routing

Noritaka Shigei† Hiromi Miyajima‡
†Dept. of Mathematics & Computer Science, Shimane University
‡Dept. of Electrical and Electronics Engineering, Kagoshima University
†Nishikawatsu 1060, Matsue, 690-8504 Japan
Tel: +81-852-32-6519, Fax: +81-852-32-6489
E-mail: shigei@cis.shimane-u.ac.jp

Abstract: This paper describes reconfiguration methods for processor arrays as a fault tolerance technique at the fabrication time. First, we show that any method based on a conventional idea cannot achieve optimal in reconfigurability. Then, we present two types of methods based on a new idea, that is sequential routing. The one is optimal in reconfigurability, and the other is advantageous in time complexity.

1.Introduction

Building the whole of a system on a large chip is advantageous in many respects, such as operating speed and power dissipation. Fault tolerance techniques for relieving defects are necessary for realizing such a implementation, because the occurrence of defects on a large chip cannot be avoided.

Reconfiguration of processor arrays have been extensively studied as a fault tolerance technique, and some models of processor array, which have redundant hardwares, such as spare processing elements (PEs) and switching circuits, have been proposed. A processor array model based on single-and-half-track switches is one of them[1]. Several reconfiguration algorithms for the model have been proposed, and most of them are based on a concept of compensation path (CP)[1, 2]. An algorithm being optimal in reconfigurability has been given in the case of one spare row and one spare column[3]. However, in the case of two spare rows and two spare columns, any optimal algorithm has not been presented.

This study describes reconfiguration algorithms for the processor array model based on single-and-half-track switches. The object of our work is to design a reconfiguration algorithm being of high reconfigurability and moderate feasibility. Although, in this study, we consider only the case of two spare rows and two spare columns, the results would be able to easily extend to cases of spare rows and columns more than two. First, we show that, when using four straight spare lines, any method based on CP is not optimal in reconfigurability, that is there exists a reconfigurable fault pattern that these methods cannot successfully reconfigure. Second, two types of reconfiguration algorithms are presented. These algorithms are not based on the concept of CP, but the concept of sequential routing. The first type of the algorithm is optimal in reconfigurability. In the algorithm, a successful mapping is exhaustively searched in a depth-first fashion. When the size of array is $N \times N$, the time complexity of the algorithm is at most $O(9^{N \times N})$. The second type of the algorithm is not optimal in reconfigurability, but advantageous in time complexity. The time complexities of the algorithms are $O(N^2)$ and $O(9^N)$.

2.Preliminaries

In the following, we assume that any variable is an integer. $[n]$ denotes a set $\{0, 1, \cdots, n\}$.

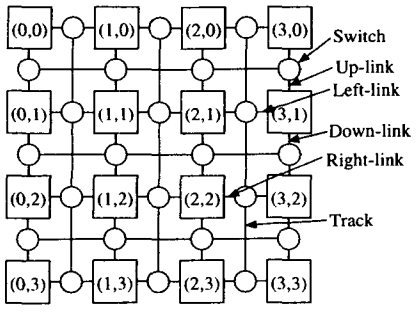
We assume that any PE on an array has the possibility of failure. In order to obtain an array of size $N \times N$ consisting of only the healthy PEs, an array of size $(N+2) \times (N+2)$ are used. A PE located at coordinates (x, y) is identified by a physical address (x, y). PEs on the leftmost column are $\{(0, y) | y \in [N+1]\}$, and PEs on the top row are $\{(x, 0) | x \in [N+1]\}$. $[x, y]$ denotes a logical address. Let $\mathcal{P} = \{(x, y) | x, y \in [N+1]\}$ be a physical array. Let $\mathcal{L} = \{[x, y] | x, y \in [N-1]\}$ be a logical array. The reconfiguration of an array is regarded as a mapping of $\mathcal{L}$ onto $\mathcal{P}$.

Definition 1 *Let $\Psi : \mathcal{L} \longrightarrow \mathcal{P}$ be an injective mapping function. The corresponding reconfiguration is said to be an* **successful reconfiguration**, *if*

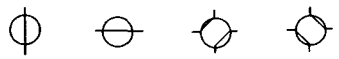
- $\forall p \in \mathcal{P} : \Psi(p)$ *is healthy,*
- *and there exists a routing such that $\forall x, y \in [N-2] : \Psi[x, y]$ is connected with $\Psi[x+1, y]$ and $\Psi[x, y+1]$.*

Definition 2 *Let $\mathcal{F}$ be an array with faulty PEs. If $\mathcal{F}$ has a mapping Ψ corresponding to a successful reconfiguration, the fault pattern of $\mathcal{F}$ is said to be* **reconfigurable**.

The processor array model based on single-and-half-track switches is shown in Fig.1[1]. The model has a single-track along every row and column (between neighboring PEs) and a switch at every intersection of local links and tracks. Each switch can be set in any of four states as shown in Fig.1.(b). Each PE has a capable of bypasses, and it can be converted into a vertical or horizontal connecting element, whether it is healthy or not. We will call links to up, down, right, left directions from each PE, up-link, down-link, right-link and



(a) The Hardware Configuration (N=3)



(b) The Switch functions

Figure 1: The reconfiguration model.

left-link, respectively. From the structure of the model, we have the following lemmas[3].

Lemma 1 *For any $x, y \in [N+1]$, an up-link of (x, y) can be connected only with a down-link of $\{(u, y-1)|u \in [N+1]\}$, and a left-link of (x, y) can be connected only with a right-link of $\{(x-1, v)|v \in [N+1]\}$.*

Lemma 2 *For any $x, y \in [N+1]$, if $\Psi[x, y] \in (x', y')$ and $\Psi[x+1, y] \in (x'', y'')$, then $x' < x''$. For any $x, y \in [N+1]$, if $\Psi[x, y] \in (x', y')$ and $\Psi[x, y+1] \in (x'', y'')$, then $y' < y''$.*

Lemma 3 *Let Ψ be a mapping function. If there exists a routing such that $\forall x, y \in [N-2] : \Psi[x, y]$ is connected with $\Psi[x+1, y]$ and $\Psi[x, y+1]$, $\forall x, y \in [N-1] : \Psi[x, y] \in \{(x+i, y+j)|i, j \in [2]\}$.*

Reconfiguration methods based on a concept of compensation path (CP) have been presented. For these methods, two spare rows and two spare columns are specified in advance. Although a spare line can be given as at most three partitioned segments, this study considers only straight spare lines [1]. A CP is defined as a horizontal or vertical straight line from a faulty PE to a spare PE. $(x, y) \rightarrow (u, v)$ denotes a CP from (x, y) to (u, v). Once a CP having no faulty PE on itself is given to a faulty non-spare PE, the faulty PE is compensated by shifting PEs along the CP. However, some pairs of CP cannot compensate the faulty PEs simultaneously; for example intersecting CPs. Therefore, a reconfiguration based on CP can be regarded as an assignment of CPs to faulty non-spare PEs. When using only straight spare lines, any method based on CP is not optimal in reconfigurability. We have the following proposition.

[1]The number of the combination is, for straight lines, $O(N^4)$ and, for partitioned lines, $O(N^{10})$. We expect that, when using partitioned spare lines, the reconfiguration based on CP is optimal in reconfigurability.

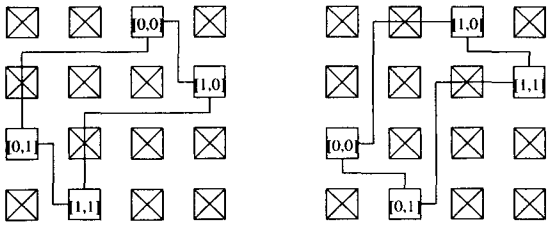


Figure 2: The mapping functions corresponding to successful reconfigurations.

Proposition 1 *When four spare lines are straight, there exists a reconfigurable fault pattern that any method based on CP cannot reconfigure successfully.*

Proof: Let us consider a reconfigurable fault pattern as shown in Fig.2. From lemma 3, we have two successful mapping functions Ψ_1 and Ψ_2 shown in Fig.2, and there is no other. To complete the proof, it is sufficient to show that, for any spare arrangement, Ψ_1 and Ψ_2 cannot be given by CPs.

For lack of space, we show only the case of Ψ_1. The case of Ψ_2 can be proved in the similar way. First, let us suppose that the 3rd (the rightmost) column is a spare line. $\Psi_1[1, 0] = (3, 1)$ requires a CP $(2, 1) \rightarrow (3, 1)$. Then, $\Psi_1[0, 0] = (2, 0)$ requires either of CPs $(0, 1) \rightarrow (2, 0)$ and $(1, 1) \rightarrow (2, 0)$. However both the CPs are slanting. Therefore the 3rd column cannot be set to a spare line. Next, let us suppose that the 2nd column is a spare line. $\Psi_1[0, 0] = (2, 0)$ requires a CP $(1, 0) \rightarrow (2, 0)$. From this, the 1st column becomes unavailable as a spare line. Since either of the 1st and 3rd columns is unavailable as a spare line, the 0th column has to be a spare line. Then, $\Psi_1[0, 1] = (0, 2)$ requires a CP $(1, 2) \rightarrow (0, 2)$. However, it causes a contradiction that $\Psi_1[1, 1] = (1, 3)$ requires a slanting CP. Therefore the 2nd column cannot be set to a spare line. Although the remaining candidates for spare lines are the 0th and 1st columns, adopting both makes $(0, 2)$ unavailable. Therefore, the 0th and 1st columns cannot be set to a spare line. It has been shown that, for any spare columns, Ψ_1 cannot be given by CPs. □

3. Reconfigurations Based on Sequential Routing

We present two types of algorithms based on sequential routing. Their basic idea is as follows:

- The images of the mapping function to the range $\mathcal{L}$ are decided one by one.
- After each decision, it is checked whether there is a valid routing for the decision.

3.1 A Reconfiguration Optimal Algorithm

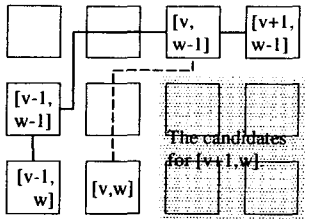
The first type is optimal in reconfigurability, and it searches for a successful mapping function in a depth-first fashion. The algorithm is given as the following

procedure RSR.

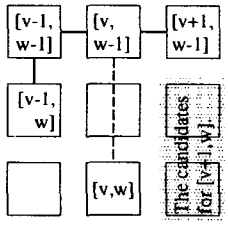
```
procedure RSR(){
  for y = 0 to 2 do
    for x = 0 to 2 do
      if (x, y) is healthy {
        if ASSIGN(0, 0, x, y) =SUCCESS
          return SUCCESS;
  return FAIL;
}
procedure ASSIGN(v, w, x, y){
  if ROUTE(v, w, x, y) =SUCCESS {
    Ψ[v, w] := (x, y);
      if v = N − 1 {
        if w = N − 1 return SUCCESS;
        v := 0; w := w + 1;
      }
      else v := v + 1;
  }
  for y = v to v + 2 do
    for x = w to w + 2 do
      if (x, y) is healthy and
          Ψ[v, w] = (x, y) satisfies lemma 2
        if ASSIGN(v, w, x, y) =SUCCESS
          return SUCCESS;
  RESTORE(v, w, x, y);
  return FAIL;
}
procedure ROUTE(v, w, x, y){
  Find a routing such that
    a choice of values assigned to Ψ in future
    are maximized.
  If a routing is found then
    set the state of switches and bypasses
      and return SUCCESS,
  otherwise return FAIL.
}
procedure RESTORE(v, w, x, y){
  Restore the state of switches and bypasses
    before ROUTE(v, w, x, y) is called.
}
```

The RSR attempts to assign a value to $\Psi[0,0]$ by calling the ASSIGN. From lemma 3, it is sufficient the attempts on $\{(x,y)|x,y \in [2]\}$. The ASSIGN examines whether there exists a valid routing for $\Psi[v,w] = (x,y)$ or not, by calling the ROUTE. If a valid routing exists, the ROUTE sets the state of switches and bypasses for $\Psi[v,w] = (x,y)$. Then, the ASSIGN attempts to assign a value to $\Psi[v+1,w]$ or $\Psi[0,w+1]$ by calling the ASSIGN recursively. The RESTORE is called by the ASSIGN, when the decision of the mapping function reaches a deadlock. After restoring the state of switches and bypasses to the previous state, the ASSIGN returns to the previous call.

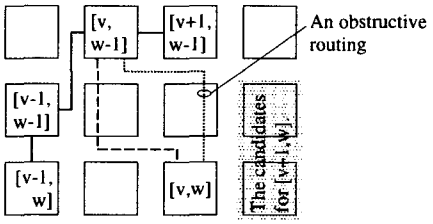
The ROUTE decides a routing for connecting $\Psi[v,w]$ with $\Psi[v-1,w]$ and $\Psi[v,w-1]$. First a vertical routing between $\Psi[v,w]$ and $\Psi[v,w-1]$ is decided, and then



(a). The routing for case (i).



(b). The routing for case (ii).



(c). The routing for case (iii).

Figure 3: The vertical routing for $\Psi[v,w]$.

a horizontal routing between $\Psi[v,w]$ and $\Psi[v-1,w]$ is decided. The routing has to maximize a choice of values assigned to Ψ in future.

Let $\Psi[v,w] = (x,y)$ and $\Psi[v,w-1] = (x^-,y^-)$. The vertical routings are classified into three cases: (i) $x < x^-$, (ii) $x = x^-$ and (iii) $x > x^-$. The routings for three cases are shown in Fig.3. In the figure, the candidates for $\Psi[v+1,w]$ are indicated by hatched squares. Any vertical routing never prevents any of the candidates from being assigned to $\Psi[v+1,w]$. In the cases (i) and (ii), the routing is established along the left and upper side of the hatched square. Since any of PEs in the hatched square is never used as a bypass element, such a routing never prevents any assignment for the candidates. In the case (iii), the routings invoked by the mapping $\Psi[v+1,w]$ and $\Psi[v+2,w]$ have to be considered. For example, an obstructive routing as shown in the Fig.3.(c) has to be avoided. Such a routing prevents from connecting $\Psi[v+1,w-1]$ and $\Psi[v+1,w]$. For the case (iii), if possible, the connection between $\Psi[v,w]$ and $\Psi[v,w-1]$ has to be routed around to avoid the area.

The horizontal routing is shown in Fig.4. Any horizontal routing never prevents any of the candidates from being assigned to $\Psi[*,w+1]$. Further, if possible, the connection between $\Psi[v,w]$ and $\Psi[v-1,w]$ has to be routed around to avoid the area.

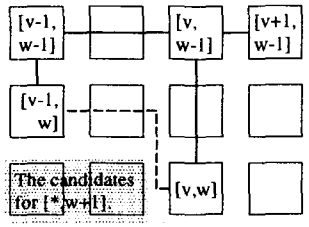


Figure 4: The horizontal routing for $\Psi[v, w]$.

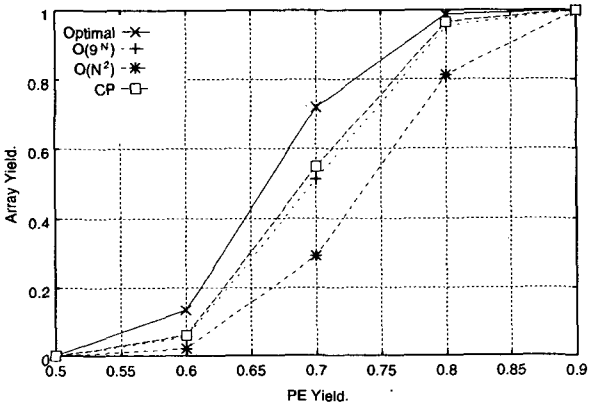


Figure 5: Simulation results (for $N = 7$)

As described above, the algorithm sequentially assigns images to Ψ, and at each assignment, the best routing is given to the assignment. If possible, the routing never reduces the candidates for the future assignments. Therefore, the algorithm examines all the combinations of the possible mappings, its reconfigurability is optimal.

Let us estimate the time complexity of the algorithm. At each assignment, at most 9 candidates are considered. The algorithm searches for a successful mapping in a depth-first fashion. Therefore, the time complexity is $O(9^{N \times N})$.

3.2 Time Efficient Algorithms

We present two types of time efficient reconfiguration algorithms based on sequential routing. Their time complexities are $O(9^N)$ and $O(N^2)$. Unlike the previous optimal one, the algorithms examines a part of the combinations of the possible mapping. For $O(9^N)$, only $\{\Psi[x, 0] | x \in [N]\}$ are assigned in a depth-first fashion. For $O(N^2)$, each target of Ψ except for $\Psi[0, 0]$ is assigned just once. These algorithms adopt the above procedures RSR, ROUTE and RESTORE, and adopt other versions instead of the above ASSIGN. For lack of space, we show only a version for $O(N^2)$.

```
procedure ASSIGN(v, w, x, y){
  if ROUTE(v, w, x, y) =SUCCESS {
    Ψ[v, w] := (x, y);
      if v = N − 1 {
        if w = N − 1 return SUCCESS;
        v := 0; w := w + 1;
      }
      else v := v + 1;
  }
  for y = v to v + 2 do
    for x = w to w + 2 do
      if (x, y) is healthy and
          Ψ[v, w] = (x, y) satisfies lemma 2
        if ASSIGN(v, w, x, y) =SUCCESS
          return SUCCESS;
        else return FAIL;
  RESTORE(v, w, x, y);
  return FAIL;
}
```

4. Simulations

To evaluate these methods in reconfigurability, we have performed computer simulations. The simulation results for $N = 7$ are summarized in Fig.5. The PE yield means the probability that a PE is healthy. The array yield means the probability that a method succeeds a reconfiguration of an array. Our three methods are compared with a method based on CP. The CP method examines all the spare arrangements in the worst case, and its time complexity is $O(N^6)$.

The reconfiguration optimal method is outstanding in reconfigurability. At PE yield 0.7, the optimal method exceeds the CP method by approximately 17%. The method of time complexity $O(9^N)$ and the CP method is almost same. The polynomial time method is the worst.

5. Conclusions

This paper presented two types of reconfiguration methods based on a new idea, that is sequential routing. The computer simulations have shown that our reconfiguration optimal method has higher reconfigurability than any other methods. And they have shown that our time efficient methods was not better than a conventional method. However, since our methods consider the routing for a reconfiguration, they would be advantageous in their implementation.

References

[1] S.Y.Kung, S.N.Jean, and C.W.Chang, "Fault-tolerant array processors using single-track switches," IEEE Trans. Comput., Vol.38, No.4, pp.501-514, April 1989.

[2] V.P.Roychowdhury, J.Bruck, and T.Kailath, "Efficient algorithms for reconfiguration in VLSI/WSI arrays," IEEE Trans. Comput., Vol.39, No.4, pp.480-489, April 1990.

[3] I.Numata, and S.Horiguchi, Tech. report of IEICE, FIIS-95-6, March 1996 (in Japanese).