A Fault Diagnosis Scheme and Its Quality Issue in Reconfigurable Array Architecture

Yung-Yuan Chen
Department of Computer Science and Information Engineering
Chung-Hua University, Hsin-Chu, Taiwan, R.O.C.
chenyy@chu.edu.tw

ABSTRACT

In this paper, we propose an efficient diagnosis scheme to detect and locate the switching network defects/faults in reconfigurable array architecture. This diagnosis scheme performs the test of switching network based on the scan path and fault intersection test methodology to locate the faults occurring in the switching network. After the diagnosis of switching network, the processing element (PE) test can then be initiated through the good switches and links. Errors in testing that cause a good switch, link or PE to be considered as a bad one is called "killing error". The issue of killing error in testing is addressed and the probability of killing error for our diagnosis technique is analyzed and shown to be extremely low. The significance of this approach is the ability to detect and locate the multiple faults in switches, links, and PEs with low testing circuit overhead, and to offer the good test quality in linear diagnosis time.

Keywords: Fault diagnosis, errors in testing, reconfigurable arrays, switching network, test quality.

1. INTRODUCTION

Reconfigurable computing systems with reconfigurable array processors become prevalent for data-parallel and computation-intensive applications [1], [2]. It is necessary to build on-chip redundancy and develop efficient reconfiguration schemes to guarantee a high yield and high reliability in large area array processors [3-5]. A complete process flow for reconfigurable array processors is illustrated in Fig. 1. As can be seen from Fig. 1, before performing the reconfiguration process, we need to identify the faulty elements in advance. Therefore, test plays an important role in reconfigurable array architectures. Most of the previous literatures in testing of reconfigurable arrays make the assumption of fault-free switching network, and only consider the faults located in the processing elements (PEs) [6-12]. A few authors take the faults arising from the switching network into account [13-15]. Since the switching network in reconfigurable arrays occupies a large portion of area, the approach oriented to end of production reconfiguration must consider the faults in switch and link locations in order to precisely locate the flawed elements. From the industry point of view, the assumption of fault-free switching network is too optimistic, and it may result in the testing quality problem. Consequently, it is imperative to test the switching network first, and identify the faults in switches and links. After the diagnosis of switching network, the part of PE test is then initiated through the good switches and links. According to this testing flow, the quality of fault diagnosis can be enhanced to a sound level.

In this study, we propose an effective diagnosis approach to detect and locate the faults occurring in switching network and PEs. The scan path and fault intersection test methodology is employed to locate the faults in switches and links. A switching network presented in [16] is used to demonstrate our diagnosis technique.

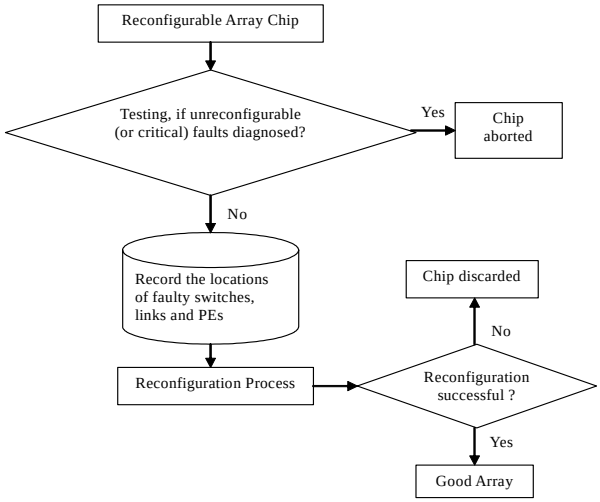


Fig. 1. A complete flow for reconfigurable array processors.

The problem of errors in testing [17] becomes more serious when the chips are getting complicated. Little or no work has been done on the problem of errors in testing for fault diagnosis scheme in reconfigurable array architecture. Errors in testing that cause a bad switch, link and PE to be considered as a good one is called "missing error", and a good switch, link and PE to be considered as a bad one is called "killing error". The missing error will result in the test quality problem for customers. The killing error during test will degrade the test quality, and may let a reconfigurable array become to be unreconfigurable. This phenomenon will lead to the yield loss and the non-return profit loss for the manufacturers. The issue of killing error in testing is addressed and the probability of killing error for our diagnosis scheme is analyzed. The probability of killing error and the diagnosis time are always contradictory. From the analysis of the killing error probability and the diagnosis time for our strategy, our approach provides a good compromise between the diagnosis time and the killing error problem.

The paper is organized as follows. In Section 2, switching network test methodology is proposed, and an example is used to demonstrate the diagnosis approach. The issue of PE test is addressed in Section 3. In Section 4, the probability of killing error is analyzed. The conclusions are drawn in Section 5.

2. SWITCHING NETWORK TEST

Switching network plays an important role in reconfigurable VLSI /WSI array systems. The switching network can be programmed to construct the diagnostic paths for switch, link and PE test. During the reconfiguration process, the switching network can be used to reconfigure the array to tolerate the fabrication

defects and in-service faults. Also, the same switching network can be exploited to extend the versatility of array topology for different application problems. Hence, it should be noticed that the testing, reconfiguration and restructuring processes in our approach share the overhead of switching network resource.

In this section, the fault model is described first. Then, a reconfigurable array and its switching network as shown in Fig. 2 and 3 [16] are used to demonstrate our test methodology.

Fault Model

The fault model we consider comprises the stuck-at, bridging, and short as well as open faults in switches and links. All multiple switch and link faults are taken into account here. Each switch has four switch states as shown in Fig. 2(b). We make the following assumptions for the switch: (1) both paths are all dead for a failed state; (2) a failed switch is defined as all states being failed.

Switching Network Architecture

The detailed switching network architecture is illustrated in Fig. 2 and 3. For the clarity, we now briefly describe the switching network architecture and the concept of fixed switch-programming scheme presented in paper [16]. Each switch as shown in Fig. 2(c) comprises a switch state scan path control circuit, two static D type flip-flops for storage of switch states, and a switch state decoder plus a transfer block. The fixed switch-programming scheme is based on the concept of scan path to scan in the switch states from I/O pads. When a switch receives its desired switch state, the switch state decoder will decode the switch state, and generate the switching type control signal to transfer block to create the desired path connection as shown in Fig. 2(b). The fixed switch-programming technique, each switch uses two-bit shift register as steering latch.

A Switch state Programming Data (SPD) path can be obtained by serially connecting the shift registers all over the switches in the path as shown in Fig. 2(c) and 3(a). From SPD I/O pads, we can scan switch-programming data into switch shift registers by clock sequences. When all programming data are loaded into their right locations, the decoder enable line is activated. Then the connection types of all switches in the programming scan path are set up. Upon finishing the procedure of switch programming, the desired interconnection paths are physically produced.

The capability of bypassing failed switches during the switch states scanning in through steering F-F paths can be accomplished by switch state scan path control circuit as demonstrated in Fig. 2(c). When a switch is faulty, the switch state scan path control circuit is used to bypass this faulty switch and to rebuild the switch state scan path. The sequence of switch programming consists of two steps. Step 1: Through switch state scan path programming pads, scan the switch status as good or faulty into the D flip-flops of switch state scan path control circuit. Then the switch state scan paths are formed. Step 2: Through switch state programming data pads, scan in the switch states, and create the desired interconnection paths.

Diagnosis Process

The switching network described in Fig. 2(a) can be partitioned into the horizontal switching network and vertical switching network as illustrated in Fig. 3(b) and 3(c). They are disjoint and can be tested concurrently. The diagnosis process consists of four testing phases: (I) Pretest phase, (II) Switching network fault detecting phase, (III) Switching network fault locating phase, and (IV) PE testing phase.

(I) Pretest phase: In this phase, from Fig. 2(c), we first test the Switch state Scan path Programming paths (SSPs) by applying the test pattern '$0^m01^{m+1}0$' [18] to detect faults in SSPs, the notation $1^m(0^m)$ to represent a sequence of m 1's(0's), m is the number of F-Fs in SSP. Each SSP contains 6(N+1) switches in horizontal switching network and 5N+2 switches in vertical switching network for an N×N array processor. The testing time for horizontal switching network SSP is 2[6(N+1)]+3 clocks and 2(5N+2)+3 clocks for vertical switching network SSP [18]. There are faulty elements existing in SSPs if testing outputs are not our expectations. In this case, we abort this array chip. Otherwise, we continue to test the Switch state Programming Data paths (SPDs).

A SPD path consists of two types of components: multiplexer (MUX), and steering F-Fs. First, we test the MUXs in a SPD path by setting the MUXs to form the path of bypassing all switch steering F-Fs in SPD. Now, the paths can be tested by the test pattern '00110'. If any faulty results are received, this array chip is discarded. Secondly, the connections of all MUXs in SPD are reversed to create the normal switch state programming data scan path which again can be tested by test pattern '$0^m01^{m+1}0$'. If erroneous outputs are observed, we continue to locate the failed steering F-Fs in those faulty switch state programming data paths. In a faulty SPD path, the steering F-F part of each switch can be diagnosed one by one by setting the path connection such that the test patterns applied only pass the steering F-F of tested switch, and bypass the steering F-Fs of other switches. In this way, we can locate the faulty steering F-Fs occurring in switches.

(II) Switching network fault detecting phase: From Fig. 2(c), each PE has two operating modes: switching network testing mode and normal operation mode. When we enable all PEs into switching network testing mode, each PE can provide the bypassing paths for switching network in testing. Then the switching network can be divided into the horizontal and vertical switching networks as shown in Fig. 3(b) and (c). In testing of switching network, we assume that DEMUX circuits are fault-free.

Although each switch has four states, from switch architecture and truth table as illustrated in Fig. 2(c), when '01', '10', '11' three states are tested, the '00' state is implicitly tested. Due to the similarity of horizontal and vertical switching network fault diagnosis, only the details of the horizontal switching network fault diagnosis are presented here. By setting up the appropriate switch states through the switch programming [16], we can construct the horizontal, vertical, right-up and right-down diagnostic paths for switches and links. Based on switch programming, when all switches of horizontal switching network are set to state '01', the Horizontal switching network Horizontal Diagnostic paths (**HHD**) and Horizontal switching network Vertical Diagnostic paths (**HVD**) are created; similarly, when switch state '10' is used, the Right-Up Diagnostic paths (**HRUD**) are formed; finally, setting the switches to state '11' can produce the Right-Down Diagnostic paths (HRDD). These types of diagnostic paths employed to test the horizontal switching network are shown in Fig. 4.

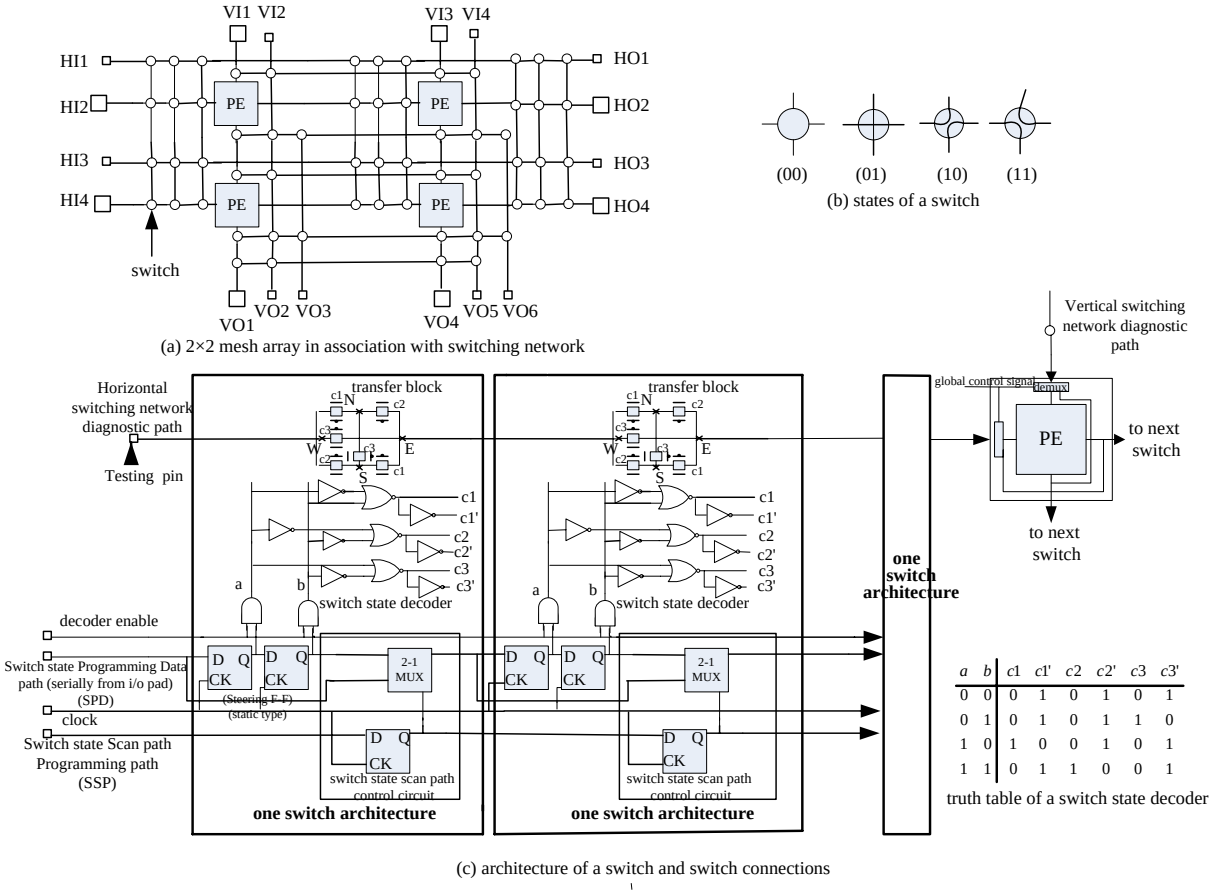


Fig. 2. Reconfigurable mesh array and its switching network.

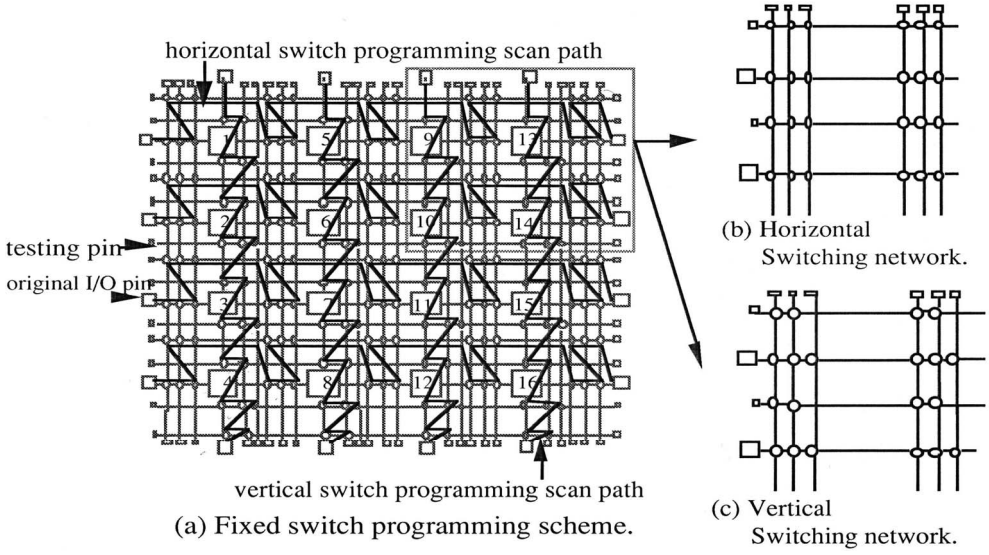
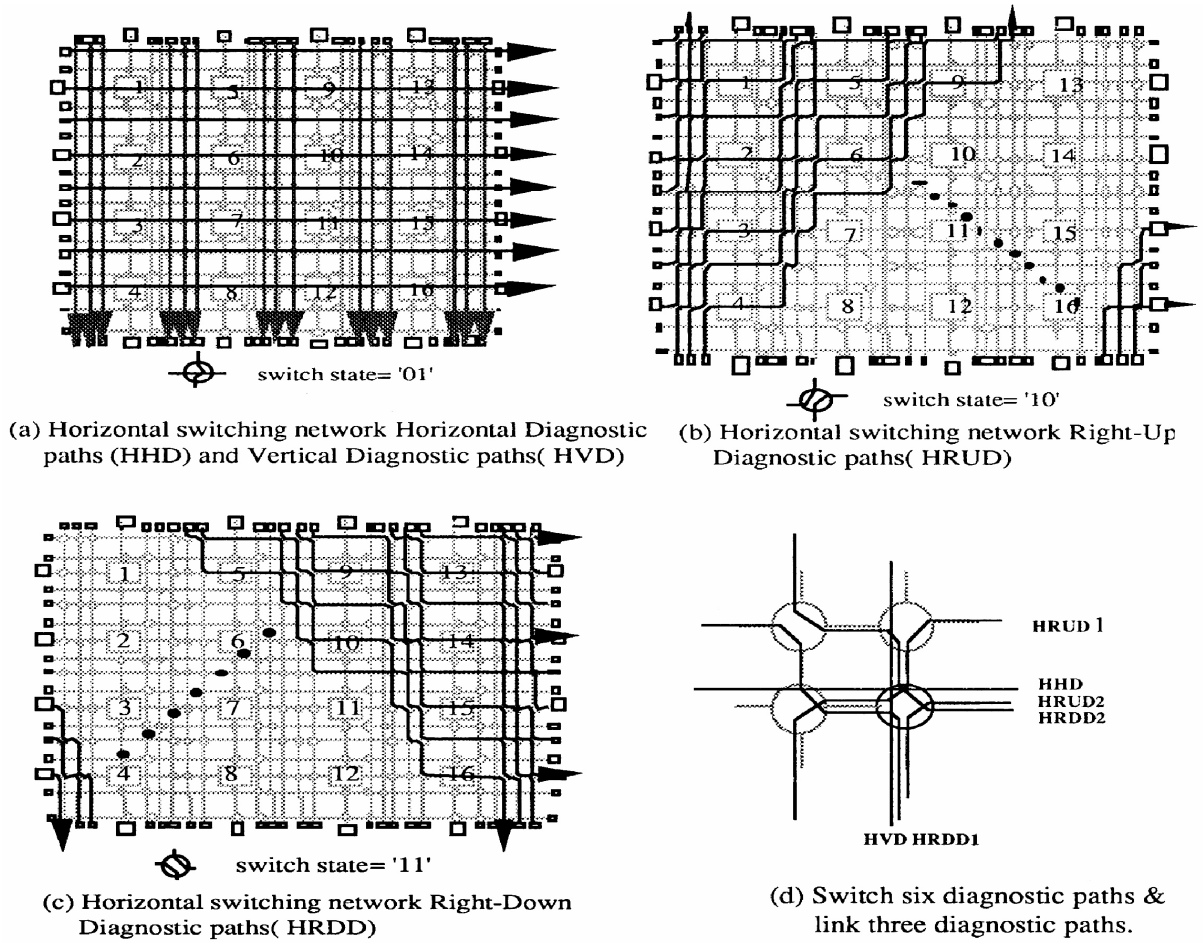


Fig. 3. Fixed switch-programming scheme and horizontal, vertical switching networks.

From Fig. 4, it is clear to see that there are six diagnostic paths passing each switch, and three diagnostic paths passing each link. The principal concept for switching network fault diagnosis is based on the fault intersection test methodology to detect and locate the faulty switches and links. The fault diagnosis process for switching network mainly consists of two phases: detecting and locating phases. Detecting phase: in this phase, the detection results from HHD, HVD, HRUD, and HRDD diagnostic paths are collected. Locating phase: according



(a) Horizontal switching network Horizontal Diagnostic paths (HHD) and Vertical Diagnostic paths(HVD)

(b) Horizontal switching network Right-Up Diagnostic paths(HRUD)

(c) Horizontal switching network Right-Down Diagnostic paths(HRDD)

(d) Switch six diagnostic paths & link three diagnostic paths.

Fig. 4. (a) HHD and HVD, (b) HRUD, (c) HRDD, (d) Switch and link diagnostic paths.

to the testing data provided from detecting phase and the fault intersection test methodology to identify the faulty switches and links.

The basic idea of fault intersection test methodology is to utilize the testing data of six diagnostic paths passing each switch and three diagnostic paths passing each link to locate the faulty switches and links. It should be noted that in accordance with the fault model described in Section 2, the six diagnostic paths passing a faulty switch must all respond the erroneous testing outputs. The fault detection algorithm of horizontal switching network is depicted as follows.

Detection Algorithm:

Step 1: Set all switches to switch state '01' to create HHD and HVD paths. Then apply the test patterns '00110' from the left side and topside to HHD and HVD paths concurrently.

Step 2: Testing outputs are observed from right and bottom sides of HHD and HVD paths. We record the results of the HHD testing outputs in Horizontal Diagnostic Matrix (**HDM**) and HVD testing outputs in Vertical Diagnostic Matrix (**VDM**). An element in the matrix represents either a switch or a link testing result. In HDM or VDM, values of the elements passed by the faulty HHD or HVD paths are all set to '1', and passed by the workable HHD or HVD paths are set to '0'.

Step 3: Program all switch steering F-Fs with '10' state, then input '00110' test patterns to all HRUD paths from left and bottom sides. Record the testing outputs from top and right sides in Right-Up Diagnostic Matrix (**RUDM**).

Step 4: Program all steering F-Fs with '11' state. Apply '00110' to all HRDD paths from top and left sides, and record the testing results from right and bottom sides in Right-Down Diagnostic Matrix (**RDDM**).

The algorithms of creating the diagnostic matrixes in detection algorithm are quite similar. Consequently, only algorithm of creation of RUDM is presented here.

RUDM Creation Algorithm:

Given an N×N mesh, the size of RUDM is (4N+1) × (6N+7), where the size of matrix can be derived easily from Fig. 2(a).

Step 1: Initialize the RUDM
for i=1 to 4N+1 **step 2**
for j=1 to 6N+7 **step 2**
RUDM (i, j)← 'u'
/* 'u' = the unused sign which is employed to assist the location of faults. */
endfor
The rest of RUDM are assigned the initial value of zero.

Step 2: for each testing output
if (the testing output is not correct)
then (In RUDM, we add one to those elements passed by this diagnostic path)

(III) Switching network fault locating phase: For any one switch, there are six diagnostic paths passing it. Hence, if a switch fails, then testing outputs of the six diagnostic paths passing it are all erroneous. Similarly, there are three diagnostic paths passing each link. Thus, if a

link fails, then the testing outputs of these three diagnostic paths passing it are incorrect. The summation of HDM, VDM, RUDM and RDDM is then performed to obtain the Diagnostic Sum Matrix (**DSM**). Based on the characteristic indicated above, the faulty switches and links in horizontal switching network can be easily located from the information of **DSM**. In DSM, if any element's value is six, then a faulty switch is identified in the corresponding location of switching network. If any element's value is three and it is between two 'u' elements then the corresponding location in switching network represents a faulty link. As can be seen, each switch has four adjacent links and each link has two adjacent switches. Therefore, each switch element in DSM has four surrounding numerical data that represent the diagnostic results of links, and each link element in DSM has two 'u' elements either located in its top and bottom or in its left and right. It should be noticed that the values of some of switch elements in DSM might be equal to three. In location of faulty links, to avoid misjudging a switch element as a faulty link, a faulty link is identified as its value is three and it is between two 'u' elements in DSM. The detailed fault location algorithm is listed below.

Location Algorithm:

for i=1 to 4N+1
  for j=1 to 6N+7
  DSM(i,j)←HDM(i,j)+VDM(i,j)+RUDM(i,j)+RDDM(i,j)
/* in above summation, if all adding items are 'u' elements then the result is still 'u'; otherwise, ignore the 'u' elements if any, and add only the items, which are numeral. */
endfor
for i=1 to 4N+1
  for j=1 to 6N+7
  if (DSM(i,j)=6) **then** corresponding location (i,j) in horizontal switching network represents a faulty switch.
  else if (DSM(i,j)=3)
      then if [(DSM(i-1,j)= 'u' **and** DSM(i+1,j)= 'u') **or** (DSM(i,j-1)= 'u' **and** DSM(i,j+1)= 'u')]
          then corresponding location (i,j) in horizontal switching network represents a faulty link.
  endif
endfor

(IV) PE testing phase: After the diagnosis of switching network, the PE testing is then initiated through the good switches and links. The PEs now are changed to the normal operation mode. The details of PE test are presented in Section 3.

Example Demonstration

Given a 2×2 mesh as displayed in Fig. 2(a), its horizontal switching network with faulty switches and links is shown in Fig. 5(a). The 'X' represents the faulty elements. The size of diagnostic matrix is 9×19. 'u' represents the unused sign in the matrix. In DSM, the location of 'u' is used to assist the identification of faulty links. It is clear to see that the concept of fault location is based on the fault diagnosis accumulation for faulty switches and links. The DSM records the total fault diagnosis accumulation. So, value six in DSM represents a faulty switch, and value three may represent a faulty link depending on its two side elements. All diagnostic matrixes for this example are given in Fig. 5(b)-(f).

Comparing the faulty switches and links located from DSM in Fig. 5(f) with original faults in Fig. 5(a), some discrepancies are observed. These differences are caused due to the problem of killing error on switches and links.

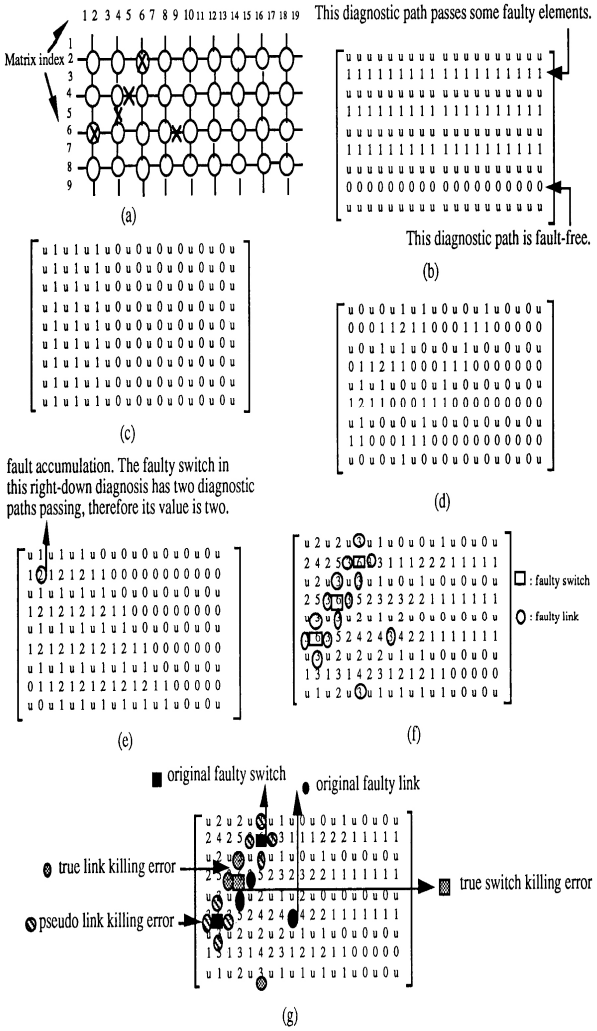


Fig. 5. (a) Example (b) HDM (c) VDM (d) RUDM (e) RDDM (f) DSM (g) Killing errors.

Definition 1: Error in testing that causes a good PE, switch and link to be considered as a bad one is called 'killing error'.

In our diagnosis approach, the killing error can be categorized into the pseudo and true killing errors.

Definition 2: The good switches and links are killed due to the errors in testing, but those switches and links won't have any chance to be utilized in the reconfiguration process at all, even though they are tested as good. This kind of killing error is called the pseudo killing error. The pseudo killing error won't affect the quality of test and reconfiguration process.

Definition 3: The good switches and links are killed due to the errors in testing, and originally these good switches and links are the useful resources for the reconfiguration process. This kind of killing error is called the true killing error. The true killing errors will affect the quality of test and reconfiguration process.

The phenomena of pseudo and true killing errors can be explained in Fig. 5(g). The good links adjacent to a faulty switch are diagnosed as faulty using our mechanism. In this situation, this is killing error for links, but the four links associated with a faulty switch are all useless during the reconfiguration process even though they are good

actually. So this kind of killing error is called the pseudo killing error and it would not affect the quality of test and reconfiguration process. The killing errors not within the type of pseudo killing error are named true killing error as shown in Fig. 5(g). The analysis of killing error for our approach is conducted in Section 4.

Test Time Analysis

The horizontal switching network and vertical switching network can be tested concurrently. The chips may have different faulty patterns. Therefore, for some chips, they are aborted in pretest phase, and the other chips may go through SSP test, SPD 'MUX' test, and SPD steering F-Fs detection and location, switching network detecting and locating phases. We omit the detailed derivation and only show the result here: $T_{sw\text{-}n}$= 144N+171 clocks. So the test time of switching network is linear, where N is the dimension of mesh array.

3. PE TESTING SCHEME

For the sake of simplicity of presentation, we here assume that the links are fault-free and only consider the switch faults in switching network in testing of PEs. The controllability and observability in PE test can be achieved through the programming of switching network. Through the switch programming, the desired testing paths for PEs can be created. The test patterns are then applied to the PEs and the testing results are observed all through the testing paths currently created. Previously, most of the papers address the issue of PE test by the assumption of fault-free switching network. This assumption, however, may give quite misleading results and degrade the quality of PE test.

The philosophy of our PE test is briefly depicted as follows. We partition the PEs into several groups based on the resources of switching network and status of switches. If a switch is faulty, it cannot be exploited in PE test. The limited numbers of input/output and the capability of switching network decide the maximum number of PEs that can be tested concurrently. A group is formed to contain the PEs, which can be tested at the same time. Therefore, if the PEs are partitioned into *n* groups and the groups are tested one by one sequentially, the procedure of PE test consists of *n* testing phases. A testing phase is defined as the time to completely test a single group.

Property 1: For an *N×N* mesh array, Fig. 6 shows the ability of switching network for horizontal part and vertical part individually. The numbers of input/output ports for horizontal switching network and vertical switching network are *2N/2N* and *2N/3N* respectively. Under the condition of fault-free switching network, the maximum number of PEs that can be tested in parallel is *2N-1*. In other words, a single testing phase can test *2N-1* PEs at most.

The explanation is described as follows. It is obvious to see that the capability of horizontal switching network supports *2N-1* input and *2N-1* output ports at most that can be exploited in a single testing phase. And, *2N* input and *2N* output ports could be utilized for vertical switching network in a single testing phase. So, due to the constraint of horizontal switching network, a single PE group can contain *2N-1* PEs at most. As a result, the minimum number of testing phases required in PE test is $\frac{N}{2}+1$ and $\frac{(N+1)}{2}$ for *N* being even and odd respectively.

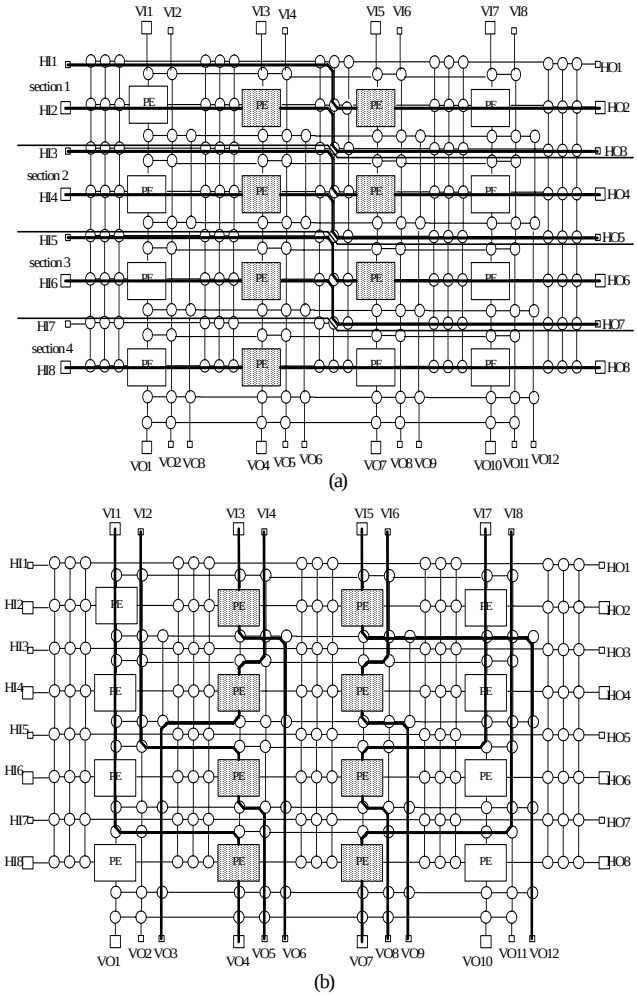


Fig. 6. Capability of switching network: (a) horizontal part, (b) vertical part.

The Approach of PE Test without Switch Faults

The PEs in an *N×N* mesh array are indexed from (1, 1) to (N, N). G(*i*) is a set of PEs for group *i*, where $1 \le i \le \frac{N}{2}+1$ for N being even, and $1 \le i \le \frac{N+1}{2}$ for N being odd. Since the test approaches for N being even and odd are similar, the following discussion only focuses on the even size array. The group partitioning algorithm is presented below.

Group Partition Algorithm: N is even

G(*i*)←Φ, for $1 \le i \le \frac{N}{2}+1$

for i = 1,…, $\frac{N}{2}$ do

begin

for j = 1,…, N do

G(i)←G(i) ∪ {PE(j, i)} ∪ {PE(j, N-i+1)}

G(i) ←G(i)–{PE(N, N-i+1)} /*to satisfy the Property 1*/

end /*Group i contains 2N-1 PEs for $1 \le i \le \frac{N}{2}$ */

for j = 1,…, $\frac{N}{2}$ do /*Group $\frac{N}{2}+1$ contains $\frac{N}{2}$ PEs*/

G($\frac{N}{2}+1$)←G($\frac{N}{2}+1$) ∪ {PE($N, \frac{N}{2}+j$)}

Example: We use Fig. 7 to demonstrate the methodologies of group partition. Fig. 7(a) is 4×4 mesh. Based on Group Partition Algorithm, the PEs are partitioned into three groups that are listed below.

G(1) = {(1,1), (2,1), (3,1), (4,1), (1,4), (2,4), (3,4)}

G(2) = {(1,2), (2,2), (3,2), (4,2), (1,3), (2,3), (3,3)}

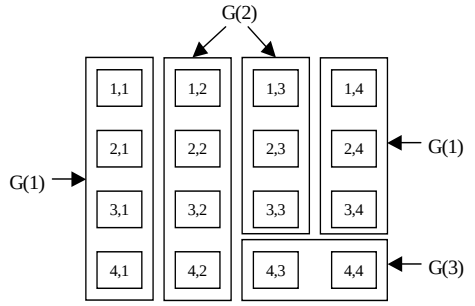
G(3) = {(4,3), (4,4)}

Fig. 7(b) is 5×5 mesh. The PEs are partitioned into three groups, which are shown as follows.

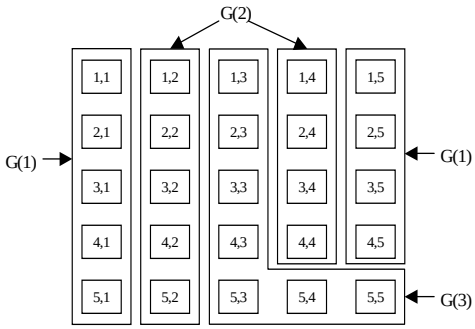
G(1) = {(1,1), (2,1), (3,1), (4,1), (5,1), (1,5), (2,5), (3,5), (4,5)}

G(2) = {(1,2), (2,2), (3,2), (4,2), (5,2), (1,4), (2,4), (3,4), (4,4)}

G(3) = {(1,3), (2,3), (3,3), (4,3), (5,3) (5,4), (5,5)} □



(a)



(b)

Fig. 7. Examples of PE group partition: (a) 4×4 mesh, (b) 5×5 mesh.

The approach of PE test under the assumption of fault-free switching network is presented next. Then, the effect of switch faults on PE test will be addressed and the technique of PE test with switch faults will be developed by the modification of the approach of PE test without switch faults.

PE Test Algorithm: with the assumption of fault-free switching network;

for i = 1,…, $\frac{N}{2}+1$ do

begin

Group partition: use Group Partition Algorithm to form group G(i);

Routing: through the switching network, establish the testing paths for each PE in G(i) from input/output ports in mesh array.

PE test: apply the complete set of PE test patterns one by one to each PE in G(i) and observe the testing results.

end

The Approach of PE Test with Switch Faults

The impacts of switch faults on PE test and the method to cope with the impacts are described as follows:

- No impact: A switch fault is not located in the switches of testing paths;
- Path-length impact: A switch fault results in the testing path longer than the path length of no switch faults;
- Routing-critical impact: As can be seen from Fig. 8, switch faults occur in the routing-critical points such that some of testing paths of the PEs in group G(i) can not be established during the testing phase of G(i). As a consequence of this impact, not all PEs in G(i) can be tested concurrently. This situation is caused due to the resource constraint of the switching network. The untested PEs will be postponed to the later testing phases. This phenomenon may induce the extra testing phases. To minimize the effect of this phenomenon, we will try to allocate the PEs from the adjacent untested group to replace the PEs that can not be tested in the testing phase of group G(i) if possible.
- Key-point impact: A switch fault is called a key-point fault of PE if the fault falls on any one of PE associated four I/O switches as shown in Fig. 8. Under the circumstances, the PE is untestable forever, and we will try to find a PE from the adjacent untested group to substitute it if possible. A PE associated with the key-point switch faults cannot be utilized during the reconfiguration process no matter what the status of this PE. Therefore, we do not need to test this kind of PEs in PE testing procedure.

Test Time Calculation

The horizontal and vertical test paths of a PE as shown in Fig. 6 each consists of three segments described as follows. The first segment is from input ports of array to the input pins of PE; the second part is PE itself and the last section is from the output pins of PE to the output ports of array. A test pattern is applied to the PE and the test results are observed all through the test paths. The delay time of a PE test pattern is the propagation delay through the test paths.

The delays of the first segment and the last segment require the consideration of the effect of switch delays and the length of the links. However, in our analysis, for simplicity of demonstration, we do not count the length of switching network channel in the test path. We assume that the length of PE edge dominates the delay among the interconnection line of the test path. The following notations are used in the derivation of the PE test time expressions.

- D_{sw}: delay associated with a switch;
- D_{ud}: Propagation delay of a signal along a unit-distance wire where a unit-distance is defined as the length of PE edge by the assumption of PE as square type;
- D_{PE}: delay time associated with a PE;
- M: number of test patterns for a PE;
- K: number of test phases;
- A_i: number of unit-distance passed in the longest test path of the i^{th} test phase, where $1 \le i \le K$.
- B_i: number of switches passed in the longest test path of the i^{th} test phase, where $1 \le i \le K$.

The delay time of a test pattern in the i^{th} test phase can be evaluated by

$$D_{PE} + (D_{ud} \times A_i) + (D_{sw} \times B_i)$$

Therefore, the test time required for the i^{th} test phase can be written as

$$M \times (D_{PE} + (D_{ud} \times A_i) + (D_{sw} \times B_i)) \qquad (1)$$

As a consequence of (1), the total PE test time for a mesh array can be expressed as

$$M \times (D_{PE} \times K + D_{ud} \times (A_1 + ... + A_i + ... + A_K) + D_{sw} \times (B_1 + ... + B_i + ... + B_K)) \qquad (2)$$

In the expression (2), the parameters of M, D_{PE}, D_{ud} and D_{sw} are design-related data, which can be acquired during the design process; for other parameters K, A_i and B_i, for $1 \le i \le K$ in above expression, the simulation approach is employed to obtain those data. The simulation results are offered and explained next.

Simulation Results

The simulation tool is based on the Monte Carlo simulation to generate a huge number of switch fault patterns to obtain the numbers of test phases K and A_i, B_i for $1 \le i \le K$. Table 1 illustrates the simulation results for several sizes of mesh array and various switch yields Y_{sw}. The notations A_T and B_T used in Table 1 represent the terms of $A_1 + ... + A_i + ... + A_K$ and $B_1 + ... + B_i + ... + B_K$ in expression (2), respectively. P is the probability of the number of test phases K demanded in PE test. The results of the simulation show that our approach performs quite well and no extra test phase is needed if a switch yield is higher than or equal to 0.999 for the size of mesh array from 4×4 to 16×16. When the yield of a switch is lower than 0.999, our scheme may need extra testing phases. The simulation results indicate that one extra testing phase is required at most and even in poor yield of a switch at 0.993, the occurring probability of an extra testing phase is still less than 35%. Clearly, the test time increases as the switch yield decreases.

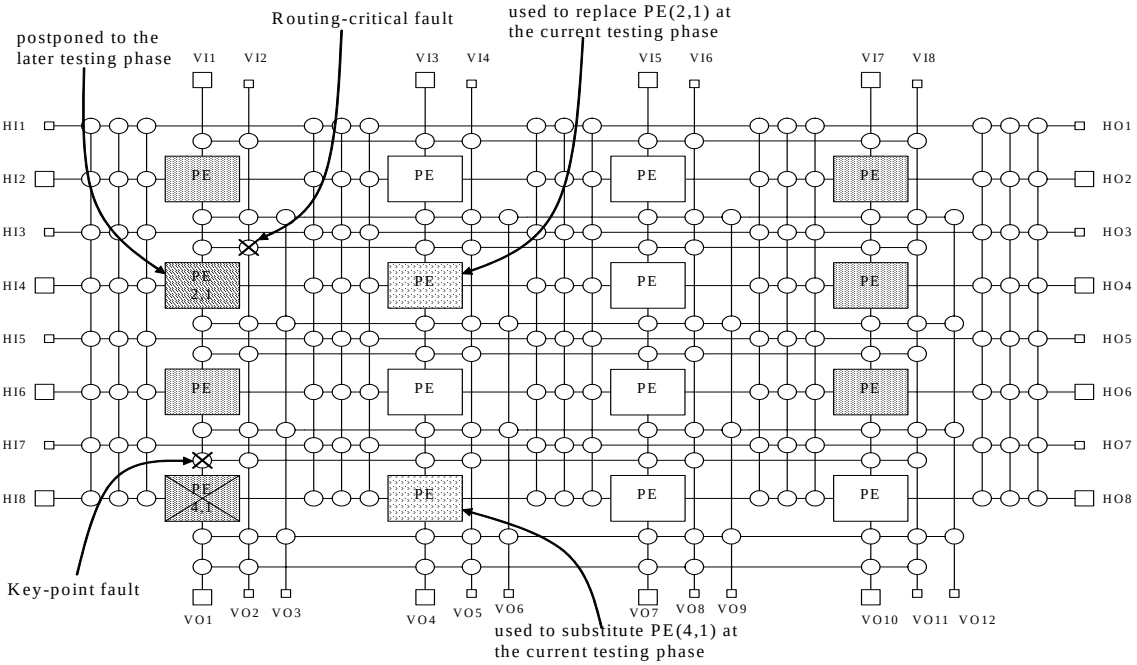


Fig. 8. The impacts of switch faults in PE test.

	Y_{sw}=1.0	Y_{sw}=0.9994	Y_{sw}=0.999	Y_{sw}=0.996	Y_{sw}=0.993
4×4 mesh	K=3 A_T=23 B_T=53	K=3 A_T=23.5~24 B_T=55~56	K=3 A_T=24~24.5 B_T=57~58	K=3 A_T=26~26.5 B_T=63~64	K=3, P=97% K=4, P=3% A_T=28~28.5, B_T=70~71
8×8 mesh	K=5 A_T=89 B_T=186	K=5 A_T=89~89.5 B_T=189	K=5 A_T=90~90.5 B_T=192	K=5 A_T=92.5 B_T=198	K=5, P=83% K=6, P=17% A_T=98~98.5, B_T=210
12×12 mesh	K=7 A_T=199.5 B_T=396	K=7 A_T=200.5 B_T=399	K=7 A_T=202 B_T=403	K=7, P=69% K=8, P=31% A_T=213, B_T=438	K=7, P=68.6% K=8, P=31.4% A_T=214.5, B_T=441
16×16 mesh	K=9 A_T=351.5 B_T=680	K=9 A_T=352.5 B_T=685	K=9 A_T=354 B_T=691	K=9, P=66% K=10, P=34% A_T=369.5, B_T=745	K=9, P=65.9% K=10, P=34.1% A_T=371, B_T=751

Table 1: Simulation results.

4. THE ANALYSIS OF KILLING ERROR

We first explain in what conditions that the switches and links will be killed during our diagnosis process. First of all, a switch ***s*** or link ***l*** is good and secondly, the six diagnostic paths passing switch ***s*** or the three diagnostic paths passing link ***l*** are all diagnosed as faulty. In other words, the six diagnostic paths passing switch ***s*** or the three diagnostic paths passing link ***l*** all have at least one faulty element occurring in each of diagnostic paths respectively. One thing should be pointed out that the probabilities of true killing error of switch and link vary with the locations of switch and link in switching network. For switches or links in different locations, they may have various numbers of switch and link components in their diagnostic paths. Obviously, a diagnostic path, which comprises a more number of switches and links, has a higher probability for at least one faulty element arising in this diagnostic path. The worst probabilities of switch and link true killing errors are derived to show that the probability of killing error for our scheme is quite low. Clearly, the probability of killing error is the summation of the probability of true killing error and the probability of pseudo killing error.

Lemma 1: The approximated worst probability of link true killing error P_{ltk} can be written as

$$P_{ltk} = P_{lk} - P_{lpk}$$

$$P_{lk} = (Y_{\ln k} Y_{sw}^2)(1 - Y_{sw}^{3N+1} Y_{\ln k}^{3N+3})(1 - Y_{sw}^{3N+2} Y_{\ln k}^{3N+4})^2$$

$$P_{lpk} = Y_{\ln k} Y_{sw}^2 [(1 - Y_{\ln k})^3 + \binom{3}{2}(1 - Y_{\ln k})^2 Y_{\ln k}(1 - Y_{sw}) + \binom{3}{2}(1 - Y_{\ln k}) Y_{\ln k}^2 (1 - Y_{sw})^2 + Y_{\ln k}^3 (1 - Y_{sw})^3]$$

where P_{lk} is the probability of link killing error, and P_{lpk} is the probability of link pseudo killing error; Y_{sw} and Y_{lnk} are the yield of a switch and a link respectively; N is the dimension of mesh array.

Lemma 2: The approximated worst probability or upper bound of switch true killing error P_{stk} can be expressed as

$$P_{stk} = P_{sk} - P_{spk}$$

$$P_{sk} = Y_{sw}(1 - Y_{sw}^{2N-3} Y_{\ln k}^{2N-1})(1 - Y_{sw}^{3N} Y_{\ln k}^{3N+2})(1 - Y_{sw}^{3N+1} Y_{\ln k}^{3N+3})^4$$

$$P_{spk} = Y_{sw}\{Y_{sw}^4 [(1 - Y_{\ln k})^4 + \binom{4}{3}(1 - Y_{\ln k})^3 Y_{\ln k}] + \binom{4}{3}(1 - Y_{sw}) Y_{sw}^3 [(1 - Y_{\ln k})^4 + \binom{4}{3}(1 - Y_{\ln k})^3 Y_{\ln k} + 3(1 - Y_{\ln k})^2 Y_{\ln k}^2] + \binom{4}{2}(1 - Y_{sw})^2 Y_{sw}^2 [(1 - Y_{\ln k})^4 + \binom{4}{3}(1 - Y_{\ln k})^3 Y_{\ln k} + 5(1 - Y_{\ln k})^2 Y_{\ln k}^2 + 2(1 - Y_{\ln k}) Y_{\ln k}^3] + \binom{4}{3}(1 - Y_{sw})^3 Y_{sw} + (1 - Y_{sw})^4\}$$

where P_{sk} is the probability of switch killing error, and P_{spk} is the probability of switch pseudo killing error.

Lemma 3: The worst probability of PE true killing error P_{petk} can be calculated by

$$P_{petk} = \{\sum_{p=0}^{4}\sum_{q=0}^{4} [\binom{4}{p}(Y_{sw}(1 - P_{sk}))^p (P_{stk})^{4-p}][\binom{4}{q}(Y_{\ln k}(1 - P_{lk}))^{4-q} (P_{ltk})^q] - (Y_{sw}(1 - P_{sk}))^4 (Y_{\ln k}(1 - P_{lk}))^4\} \times Y_{pe}$$

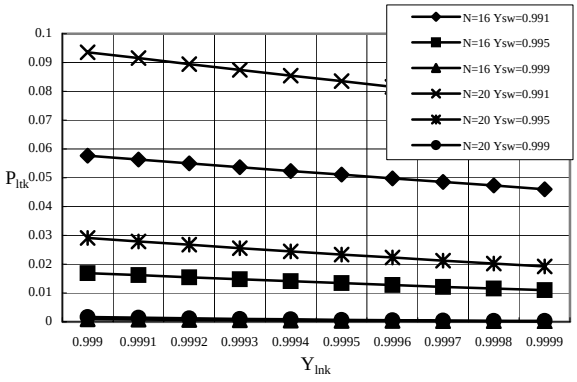
where Y_{pe} is the yield of a PE.

Fig. 9 illustrates the P_{ltk}, P_{stk} and P_{petk} for 16×16 and 20×20 mesh arrays. It should be emphasized that the probabilities of true killing error shown are the worst probability for switch, link and PE. From Fig. 9, we can see that for larger arrays with lower link and switch yield, the P_{ltk}, P_{stk} and P_{petk} increase rapidly. Interestingly, while Y_{pe} increases, the P_{petk} rises too. The reason is that under the same Y_{sw} and Y_{lnk}, more number of good PEs will be truly killed for higher Y_{pe} of arrays than the lower Y_{pe} of arrays. Therefore, on the average, when Y_{pe} increases, each PE has a higher probability to be truly killed during the diagnosis process. This indicates that while Y_{pe} increases, we also require raising the Y_{sw} and Y_{lnk} to guarantee the test quality. Table 2 lists the number of switches and PEs that are truly killed for various probabilities of switch yield, where Y_{lnk} = 0.9995 and Y_{pe} = 0.8. According to Fig. 9 and Table 2, we observe that the probability of true killing error is quite sensitive to switch yield. To obtain a good test quality, we must pay attention in the design of switching network.

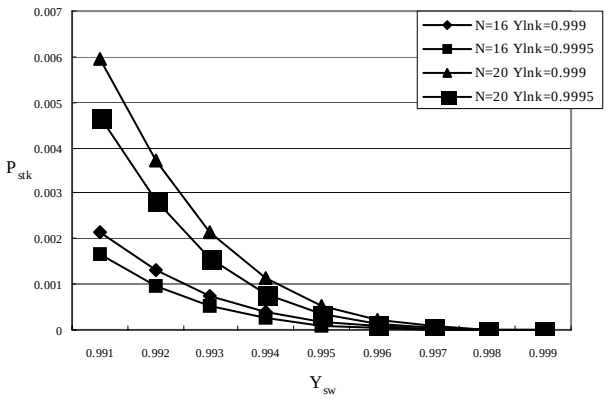
As can be seen from Fig. 9, the P_{ltk}, P_{stk} and P_{petk} are very low while Y_{sw} is 0.999 or higher. The numbers of switches and PEs that are truly killed also approach to zero as shown in Table 2 when Y_{sw} is 0.999. It is evident that the probability of killing error in our diagnosis scheme is quite low while $Y_{sw} \geq 0.999$.

	Y_{sw}= 0.991	Y_{sw}= 0.995	Y_{sw}= 0.999
16×16 (switch)	5	0.3	0
20×20 (switch)	21	1.5	0
16×16 (PE)	30	8	0.2
20×20 (PE)	75	22	0.7

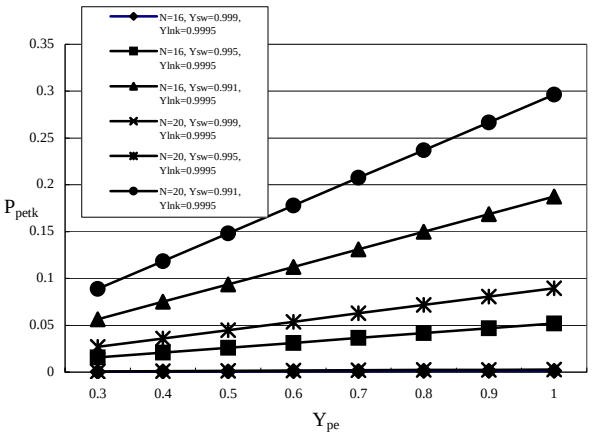
Table 2: Number of switches and PEs truly killed for various switch yields.



(a)



(b)



(c)

Fig. 9. The probability of true killing error: (a) link, (b) switch, (c) PE.

5. CONCLUSIONS

An effective fault diagnosis approach for reconfigurable VLSI/WSI array processors is presented. The issue of killing error in testing is addressed and the probability of killing error for our diagnosis scheme is analyzed and shown to be extremely low. The significance of this approach is the ability to detect and locate the multiple faults in switches, links, and PEs with low testing circuit overhead, and to offer the good test quality in less diagnosis time. It should be noted that our diagnosis approach can be applied to other types of switching networks easily. Moreover, the diagnosis concept presented here can be extended with no difficulty to cope with a more general fault model for the switch, which allows for partially failed in a switch. In this paper, for the simplicity of presentation, we employed the fault model described in Section 2 to demonstrate our diagnosis idea.

Previously, most of the reconfiguration techniques assume that the test has been performed and the faulty elements have been identified exactly. The assumption of perfect testing process is unrealistic. Errors in testing inevitably exist and will affect the accuracy of predicted yield for reconfigurable large area arrays. Hence, we need to consider the effect of errors in testing on reconfiguration process and final estimated yield. The phenomenon of errors in testing will degrade the test quality and cause the yield loss and the non-return profit for the manufacturers. Therefore, it is imperative that we need to guarantee the test quality of the diagnosis scheme developed. As can be seen from the previous analysis, our diagnosis approach can achieve the good quality of test with low-test time. The proposed methodology can be applied to the reconfigurable computing systems with reconfigurable array processors or FPGA-based reconfigurable systems to enhance the system yield and reliability.

6. REFERENCES

[1] H. Singh et al., "MorphoSys: An Integrated Reconfigurable System for Data-Parallel and Computation-Intensive Applications," *IEEE Trans. On Computers*, vol. 49, no. 5, pp. 465-481, May 2000.

[2] K. Bondalapati and V. K. Prasanna, "Reconfigurable Computing Systems," *Proceedings of the IEEE*, vol. 90, no. 7, pp1201-1217, July 2002.

[3] R. Negrini, M. G. Sami and R. Stefanelli. *Fault-Tolerance through Reconfiguration of VLSI and WSI Arrays.* The MIT Press, 1989.

[4] L. E. LaForge, "Configuration of Locally Spared Arrays in the Presence of Multiple Fault Types," *IEEE Trans. on Computers*, vol. 48, no. 4, pp. 398-416, Apr. 1999.

[5] M. Fukushi and S. Horiguchi, "A Self-Reconfigurable Hardware Architecture for Mesh Arrays Using Single/Double Vertical Track Switches," *IEEE Trans. on Instrumentation and Measurement*, vol. 53, no. 2, pp. 357-367, April 2004.

[6] Y. H. Su, M Cutler and M. Wang, "Self-Diagnosis of Failures in VLSI Tree Array Processors," *IEEE Trans. On Computers*, vol. 40, no. 11, pp. 1252-1257, Nov. 1991.

[7] D. M. Blough and A. Pelc, "Diagnosis and Repair in Multiprocessor Systems," *IEEE Trans. on Computers*, vol. 42, no. 2, pp. 205-217, Feb. 1993.

[8] J. Salinas and F. Lombardi, "Diagnosis of Reconfigurable Two-Dimensional arrays Using a Scan Approach," *6th annual IEEE Int'l Conf. on Wafer Scale Integration*, pp. 179-187, 1994.

[9] K. C. Wang and J. W. Lin, "Integrated Diagnosis and Reconfiguration Process for Defect Tolerant WSI Processor arrays," *6th annual IEEE Int'l Conf. on Wafer Scale Integration*, pp. 198-207, 1994.

[10] S. Goldberg and S. J. Upadhyaya, "Utilizing Spares in Multichip Modules for the Dual function of Fault Coverage and Fault Diagnosis," *IEEE Int'l Workshop on Defect and Fault Tolerance in VLSI Systems*, pp. 234-242, 1995.

[11] F. J. Meyer, F. Lombardi and J. Zhao, "Good Processor Identification in Two-Dimensional Grids," *IEEE Int'l Symposium on Defect and Fault Tolerance in VLSI Systems*, pp. 348-356, 1999.

[12] S. Goldberg, S. J. Upadhyaya and W. K. Fuchs, "Recovery Schemes for Mesh Arrays Utilizing Dedicated Spares," *IEEE Trans. on Reliability*, vol. 53, no. 4, pp. 445-451, Dec. 2004.

[13] S. Y. Kuo, K. C. Wang, "Fault Diagnosis in Reconfigurable VLSI and WSI Processor Arrays," *Journal of VLSI Signal Processing* 2, pp. 173-187, 1990.

[14] A. Jain, B. Mandava, J. Rajski and N. C. Rumin, "A Fault-Tolerant Array Processor Designed for Testability and Self-Reconfiguration," *IEEE Journal of Solid-State Circuits*, vol. 26, no. 5, May 1991.

[15] S. Rangarajan, D. Fussell, M. Malek, "Efficient Fault Diagnosis of Switches in Wafer Arrays," *4th annual IEEE Int'l Conf. on Wafer Scale Integration*, pp. 341-351, 1992.

[16] Y. Y. Chen, C. H. Cheng and Y. C. Chou, "An Effective Reconfiguration Process for Fault-Tolerant VLSI/WSI Array Processors," EDCC-1, pp. 421-438, Oct. 1994.

[17] R. H. Williams, C. F. Hawkins, "Errors in Testing," *Int'l Test Conf.*, pp. 1018-1027, 1990.

[18] K.J. Lee, M.A. Breuer, "A Universal Test Sequence for CMOS Scan Registers," *IEEE Custom Integrated Circuit Conf.*, pp. 28.5.1-28.5.4, 1990.

Received: Nov 2005. Accepted: Feb. 2006.