Des Autom Embed Syst (2011) 15:111–132
DOI 10.1007/s10617-011-9074-6

A scalable built-in self-recovery (BISR) VLSI architecture and design methodology for 2D-mesh based on-chip networks

Kun-Chih Chen · Shu-Yen Lin · Wen-Chung Shen · An-Yeu (Andy) Wu

Received: 20 May 2010 / Accepted: 3 March 2011 / Published online: 5 April 2011

Abstract On-Chip Networks (*OCN*s) have been proposed to solve the complex on-chip communication problems. In Very Deep-Submicron era, *OCN* will also be affected by faults in chip due to technologies shrinking. Many researches focused on fault detection and diagnosis in *OCN* systems. However, these approaches didn't consider faulty *OCN* system recovery. This paper proposes a scalable built-in self-recovery (BISR) design methodology and corresponding Surrounding Test Ring (*STR*) architecture for 2D-mesh based *OCNs* to extend the work of diagnosis. The BISR design methodology consists of STR architecture generation, faulty system recovery, and system correctness maintenance. For an $n \times n$ mesh, *STR* architecture contains one controller and $4n$ test modules which are formed as a ring-like connection surrounding the *OCN*. Moreover, these test modules generate test patterns for fault diagnosis during warm-up time. According to these diagnosis results, the faulty system is recovered. Finally, this paper proposes a fault-tolerant routing algorithm, Through-Path Fault-Tolerant (*TP-FT*) routing, to maintain the correctness of this faulty system. In our experiments, the proposed approach can reduce 68.33∼79.31% unreachable packets and 4.86∼23.6% latency in comparison with traditional approach with 8.48∼13.3% area overhead.

Keywords On-Chip Networks (*OCN*) · Built-In Self-Recovery (*BISR*) · Surrounding Test Ring (*STR*) · Design-for-Testability (*DfT*) · Through-Path Fault-Tolerant (*TP-FT*) routing

K.-C. Chen (✉) · S.-Y. Lin · W.-C. Shen · A.-Y. Wu
Graduate Institute of Electronics Engineering and Department of Electrical Engineering, National Taiwan University, Taipei, 106, Taiwan, R.O.C.
e-mail: kunchih@access.ee.ntu.edu.tw

S.-Y. Lin
e-mail: linyan@access.ee.ntu.edu.tw

W.-C. Shen
e-mail: thinking@access.ee.ntu.edu.tw

A.-Y. Wu
e-mail: andywu@cc.ee.ntu.edu.tw

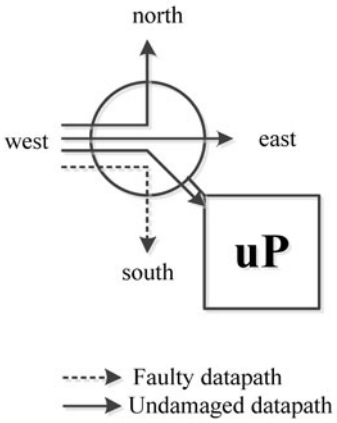

Fig. 1 Example of partial faulty router

1 Introduction

Multicore system is a popular design in recent years. For the communication of multicore systems, On-Chip Networks (*OCNs*) have been proposed to overcome wiring complexity and serious crosstalk problems of traditional wire/bus-based interconnection [1]. As CMOS technology scaling down to Very Deep-Submicron (VDSM), problems of breakdown and failure in devices and interconnections are more serious [2]. Hence, fault-tolerant approaches must be considered in *OCNs*.

In a generic *OCN* with many replicated processors and routers, a common fault-tolerant approach is to deactivate the faulty processors and remap tasks on remaining ones in software application. However, the processor-deactivate approach cannot handle faulty routers in the *OCN*. Faulty routers cause the systems unusable unless the *OCN* can be reconfigured to work correctly.

In order to test and diagnose the *OCN* system, two methods are proposed: *Built-in self-diagnosis (BISD-based)* mechanisms and *Design-for-Test (DfT-based)* approaches. In previous work, many researches focus on *BISD* of *OCNs* [3–5]. However, faults may only influence certain functions of a faulty router, and undamaged parts can still work correctly if faulty parts are detected and located as shown in Fig. 1. In order to support aforementioned features, each *OCN* must embed *BISD* circuits [5], but *BISD*-based approach causes a significant area overhead. To overcome this critical area overhead, *DfT*-based approaches are proposed in [6–10]. However, these approaches suffer from long test cycles while the scale of *OCN* grows. Furthermore, test frequency may be dominated by external test sources (*e.g.*, Automatic Test Equipment) or I/O pins, since these approaches cannot work without test equipment.

In addition to area overhead in *BISD*-based and long test cycle in *DfT*-based approaches, faulty OCN system recovery is not considered in these approaches. To extend the works of diagnosis, this paper proposes:

(1) A scalable Built-In Self-Recovery (*BISR*) design methodology

The design methodology consists of a *BISR* architecture generation, called Surrounding Test Ring (*STR*) architecture generation, faulty system recovery, and normal operation as shown in Fig. 3. In the proposed *BISR* design methodology, the 20-path router model [5] and test datapaths configuration [10] are used to diagnose the faulty *FIFOs* and *MUXs* in a router during the warm-up time before normal operations in detail, and

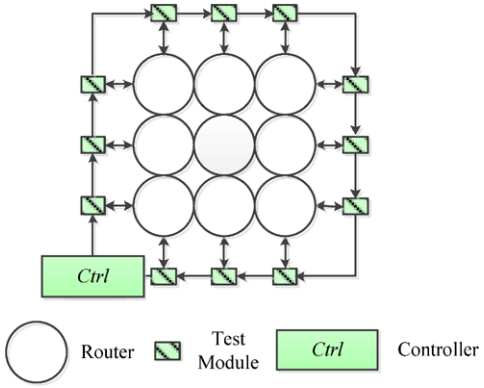

Fig. 2 The proposed surrounding test ring approach with *BISR* capability

the two technologies will be both described in later. Because *FIFOs* and *MUXs* dominate most of the area in a router, only these components are considered in the fault model.

(2) Surrounding Test Ring (*STR*) VLSI architecture

STR architecture is a corresponding architecture of *BISR* methodology. For an $n \times n$ mesh, the architecture consists of one controller and $4n$ test modules which are formed as a ring-like connection surrounding the *OCN* as shown in Fig. 2. Compared with traditional *BISD*-based approaches [5], the area overhead is significantly reduced.

(3) Through-Pass Fault-Tolerant (*TP-FT*) Routing Algorithm

In conventional fault-tolerant routing algorithm [11–13], packets will detour the faulty router which is entirely closed as one fault is detected. It causes performance degradation because heavy traffic congestion rounds the faulty router. With the support of 20-path router model *TP-FT* routing applies the fault-free datapaths in the faulty routers, which is shown in Fig. 1, and isolates the faulty *FIFOs* and *MUXs* to improve faulty system performance. The proposed TP-FT algorithm is applied to reduce 68.33∼79.31% unreachable packets which mean the packets that cannot reach their destination and 4.86∼23.6% latency in comparison with traditional fault-tolerant routing algorithms [11–13] when faulty nodes occur in *OCNs*.

The rest of this paper is organized as follows: In Sect. 2, we briefly review the related test mechanisms for *OCNs*. In Sect. 3, we introduce the scalable *BISR* design methodology. In Sect. 4, we describe our proposed *STR* architecture and *TP-FT* routing algorithm. Section 5, the implementations and experiments are discussed. Finally, we summarized this paper in Sect. 6.

2 Review of related works

The problems of fault detection and diagnosis in *OCNs* have been studied in various works. In comparison with traditional wire/bus-based interconnection, the architectures of mesh-based *OCNs* are more regular. As a result, many previous works proposed testing schemes and reused *OCNs* as test access mechanisms for IP testing [14–16]. However, as CMOS technology scaling down to Very Deep-Submicron (VDSM), problems of failure and breakdown also affect routers and wires on *OCNs*. Recently, many researches focus on testing in *OCNs*, and these works can be classified as *DfT-based* solutions and *BISD-based* solutions.

2.1 *DfT*-based solutions

DfT-based approaches are based on implementing design-for-testability structures (i.e., wrappers, scan chains, and dedicated hardwares). Hosseinabady *et al*. [6] proposed a wrapper with scan-chains attached to each router on *OCNs*. One of the routers is defined as a test access switch to receive test patterns from the external test source and broadcast these patterns to other routers. The wrappers compare output responses of each router on *OCN* to detect faults. Amory *et al*. [7] proposed a partial scan method on an IEEE 1500-compliant test wrapper, which applies the regularity of *OCNs*. In [8–10], Raik *et al.* proposed an external test approach, where insertion of wrappers and scan paths are not required. This method uses test datapaths configuration to detect faults in OCNs.

In general, there are some pros and cons of *DfT*-based solutions discussed in [17]. The advantages of these kinds of approaches are listed as follow:

(1) Test patterns are generated by ATPG tools, and test coverage can be evaluated directly by fault simulation.
(2) The *DfT* design is a typical step of VLSI design flow. And hence *DfT*-based design is a general solution for all kinds of components.

Moreover, *DfT*-based solutions contain some disadvantages:

(1) External test sources are required. The DfT-based design cannot work without test equipments.
(2) Test frequency may be dominated by external test sources or I/O pins. If the tester cannot reach the timing accuracy required to test the design, test frequency should be reduced and some defects may not be detected.
(3) Long test cycles are required when mesh scale increases.

2.2 *BISD*-based solutions

BISD-based approaches can test *OCNs* by embedded self-testing circuits without external test sources supported. Petersén *et al*. [3] proposed a *BISD* engine embedded in each network interface (*NI*). Each router is divided into two parts: (1) datapath and (2) controller. To test the datapaths, some deflection components are implemented in the router's datapath and directly controlled by the *NI*. Test patterns are sent by the *NI* to test all datapaths attached to the same router. Grecu *et al*. [4] proposed another *BISD* method for inter-switch links between routers. Test error detectors (*TED*) are implemented in two ends of each inter-switch link. A global test controller (*GTC*) injects test packets into *OCN*, and *TED* analyzes responds on *OCN* links. In [5], Lin *et al.* embedded *BISD* circuits in each router in *OCNs*. This approach not only detects the permanent faults in a router but also indicates the locations of fault.

There are also some pros and cons of *BISD*-based approaches discussed in [17]. The advantages of these kinds of approaches are listed as follows:

(1) Identification of faulty components is easier. *BISD* implements most test functions on chip and hence no external test equipments are needed. Defects can be found without being modeled by fault modeling on software.
(2) *BISD* design is manually applied to the design, and hence test cycle is predictable and controllable by designers.

Furthermore, *BISD*-based solutions contain some disadvantages:

(1) *BISD* design should be applied when designer know the design under test very well and specific *BISD* designs should be applied to different components. It requires additional design effort.
(2) The storage of test patterns or test pattern generator occupies chip space and effects overhead of *BISD* architecture. Both look up table and pseudo random number generators occupy chip space.

Aforementioned test mechanisms can provide the ability of fault detection for *OCNs*, and some of them can further support fault location. However, most of current works do not support fault isolation for further faulty system recovery. In order to support fault isolation, we provide a design methodology from built-in self-diagnosis (*BISD*) to built-in self-recovery (*BISR*) in warm-up time of *OCN* system. Furthermore, new router and surrounding test ring (*STR*) architecture is proposed to support the BISR mechanism.

3 A scalable *BISR* design methodology

In this section, we introduce a scalable *BISR* design methodology which contains three main phases (shown in Fig. 3):

(1) *STR* architecture generation
This is an off-line design. Because different mesh scale and input buffer depth cause different test pattern and numbers of test module to diagnose the faulty router in *OCNs*, two system parameters, mesh scale and input buffer depth, will be collected in this phase.
(2) Faulty system recovery
This phase will be executed during system warm-up time. the test module using the test patterns, which are designed and stored in advance, to diagnoses the faulty *FIFOs* and *MUXs* based on the 20-path router model (shown in Table 2) and recovers faulty system with isolation circuit to mask the faulty components.
(3) Normal operation
In this phase, the *TP-FT* routing algorithm is proposed to reduce the unreachable packets and latency in comparison with traditional fault-tolerant routing algorithms.

3.1 Generate scalable *STR* pattern

In the *STR* generation phase, the test modules and desired test patterns are automatically generated and inserted into *OCNs*. The scalable test patterns depend on the mesh scale and input buffer depths. The test patterns for faulty *MUXs* and mesh scale are discussed with the proposed *BISR* algorithm in Sect. 3.3. The test patterns for *FIFO* depths are described as follows:

Test FIFOs: To simplify the problem, we use multi-level 2-to-1 *MUXs* to implement a D-to-1 *FIFO MUX* when $D > 2$, and D represents the *FIFO* depth. For each *FIFO*, the X and X' test patterns are needed. The X and X' with different *FIFO* depths are shown in Table 1. All-0 and All-1 patterns represent test vectors (00...00) and (11...11). With these test patterns, we can guarantee each input pair of a 2-to-1 MUX is bit-reversal. It can help to distinguish the input faults of a 2-to-1 MUX easily. Using the input patterns transition from X to X' can detect the stuck-at faults (*SAFs*) in FIFO registers and FIFO MUXs by monitoring the *MUX's* output. Figure 4 illustrates an example to test 4-depth *FIFO*. First, X patterns in *FIFO* are assigned. Then, X' patterns in *FIFO* are assigned. Therefore, transitions

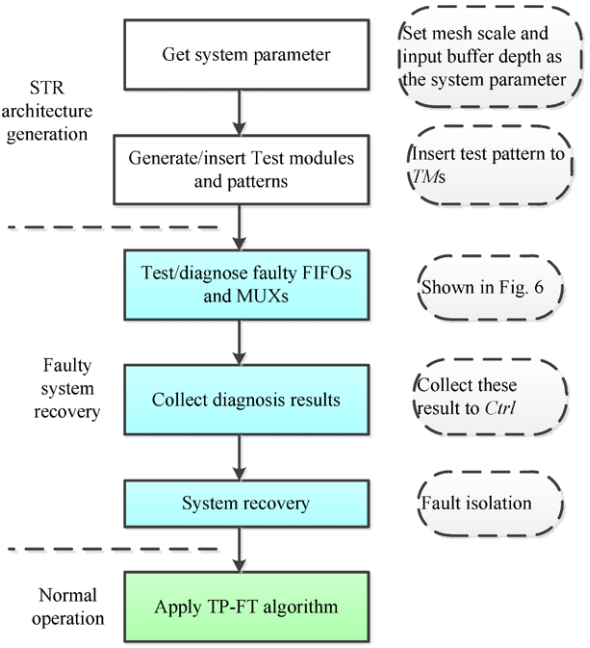

Fig. 3 The scalable *BISR* design methodology

Table 1 Pattern X and X' to test *FIFOs*

Depth	X	X'
1	{All-0}	{All-1}
2	{All-0, All-1}	{All-1, All-0}
3	{All-0, All-1, All-1}	{All-1, All-0, All-0}
4	{All-0, All-1, All-1, All-0}	{All-1,All-0, All-0, All-1}
5	{All-0, All-1, All-1, All-0, All-1}	{All-1, All-0, All-0, All-1, All-0}

in *FIFO* are observable at the *MUX's* output. If there is a stuck-at-zero (*SA0*) fault occurred in A, the output O's transition state is zero to zero rather than zero to one. By this way, this FIFO can be detected whether it is faulty or non-faulty. Because each D-flit-depth *FIFO* need to be tested using both X and X' patterns, it needs $2D$ cycles. After testing each *FIFO* depth, the read and write pointers in FIFOs need one extra cycle to be reset again. Therefore, for a D-flit-depth *FIFO*, $2D + 1$ cycles are needed to test each *FIFO* of a router. When H is denoted as the hop numbers in the critical test path and L denoted as packet delay through a router, the maximum test cycles is $(2D + 1) + H \times L$. For East, South, West, and Local Input *FIFO*, the test methods are similar.

3.2 A scalable Build-In Self-Recovery flow

Figure 2 shows the block diagram of the *STR* on a 3×3 *OCN*. The *STR* contains two kinds of components: (1) Test Modules (*TMs*) and (2) a controller (*CTRL*). The *TMs* and the *CTRL* form a ring-like connection surrounding the *OCN*. Each TM is connected to one router at the boundary of the *OCN*. By the connections between test modules and the routers, *STR*

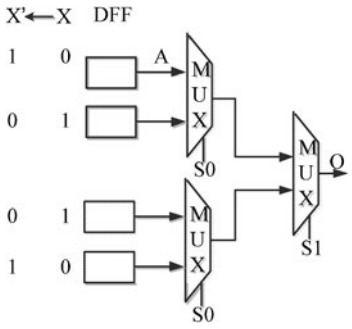

Fig. 4 Test patterns for 4-depth *FIFO*

can use the regularity of the 2D-mesh topology to test, diagnose, and isolate the impacts of the faulty *FIFOs* and *MUXs*. Besides, the architecture of *STR* is scalable because *STR* can support any size of 2D meshes.

Each *TM* can generate and receive test packets through the connections between routers and the *TMs*. Test packets transmitted between *TMs* can detect faults in the routers which are passed through. The *CTRL* controls the operations of the *TMs* and collects the test results by the connections between the *TMs* and the *CTRL*.

The recovery flow of *STR* is shown in Fig. 5. The steps are described as follows:

(1) Test faulty datapaths: *TMs* can generate and receive test packets through the connections between test modules and routers. Hence, packets transmitted between two *TMs* can detect the faulty datapaths in the routers which are passed through, as the example in Fig. 6(a).
(2) Collect test results: *CTRL* collects the test results in test modules, as the example in Fig. 6(b). The arrows with solid lines show the paths to shift the test results from the *TMs* to *CTRL*.
(3) Diagnose faulty *FIFOs* and faulty *MUXs*: *CTRL* analyzes the test results and diagnoses faulty *FIFOs* and faulty *MUXs* in *OCNs*, as the example in Fig. 6(c).
(4) Isolate faulty components in the faulty routers: *CTRL* transmits some packets to isolate the faulty components in the faulty router as the example in Fig. 6(d). These packets are defined as Fault-Isolation packets (*FI* packets).

In *STR*, the methods to transmit and receive test and *FI* packets between *TMs* are important. The methods may influence the test cycles and covered datapaths in the *OCNs*. In Sect. 3.3, the methods to test, diagnose, and isolate the faulty *FIFOs* and *MUXs* are introduced. In Sect. 3.4, *TP-FT* routing algorithm which improves the faulty *OCN* system performance is described.

3.3 Test, diagnosis, and isolation methods

As mentioned in Sect. 1, we use 20-path router model which is modeled as 20 different datapaths for each input-output pair of a router by the symbols (as shown in Table 2). In *STR*, similar to test datapaths configuration in [10], these datapaths are tested and diagnosed into 3 different categories: (1) *Turn*, (2) *Thru*, and (3) *Source/Sink* datapaths, as shown in Table 3. The methods to test and diagnose the *Turn*, *Thru*, and *Source/Sink* datapaths are discussed in Sect. 3.3.1 and 3.3.2. The method to isolate the impact of faults is discussed in Sect. 3.3.3.

Aforementioned, *CTRL* will analyze the test result and diagnose the fault *FIFOs* and *MUXs* in *OCN*. The diagnosis flow is shown in Fig. 7. First, all datapaths are set to faulty.

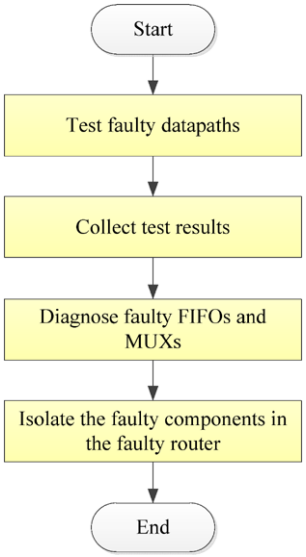

Fig. 5 The flow chart of the *STR*

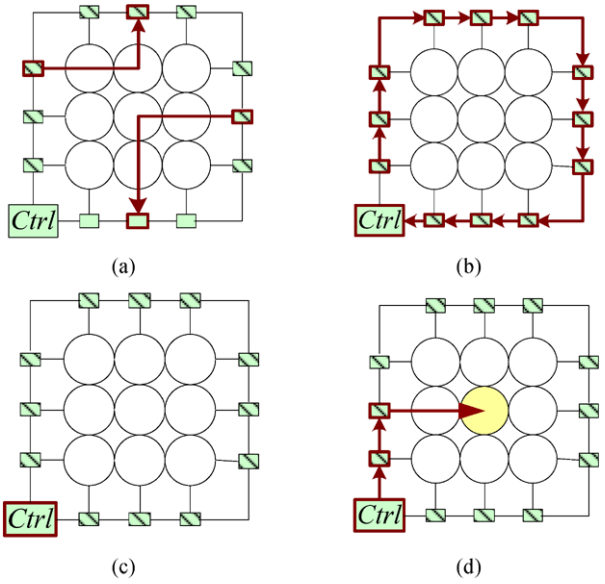

Fig. 6 The flow of the *BISR*: (**a**) the test of the faulty datapaths, (**b**) the collection of the test results, (**c**) the diagnosis of the faulty *FIFOs* and *MUXs*, and (**d**) the isolation of the faulty components in the faulty routers

Then the proposed diagnosis strategy is executed, and they will be discussed in Sect. 3.3.1 and 3.3.2. If the datapath is correct after each partial diagnosis (shown in Fig. 10 and Fig. 11), it is set to fault-free. Therefore, the datapaths which are not set to fault-free are seen as faulty datapath after the proposed diagnosis strategy.

3.3.1 Test and diagnosis methods for Thru and Turn datapaths

For testing the *Thru* and *Turn* datapaths, the *TMs* at the north and south (east and west) sides transmit the test packets to the *TMs* in the east and west (north and south) sides of the

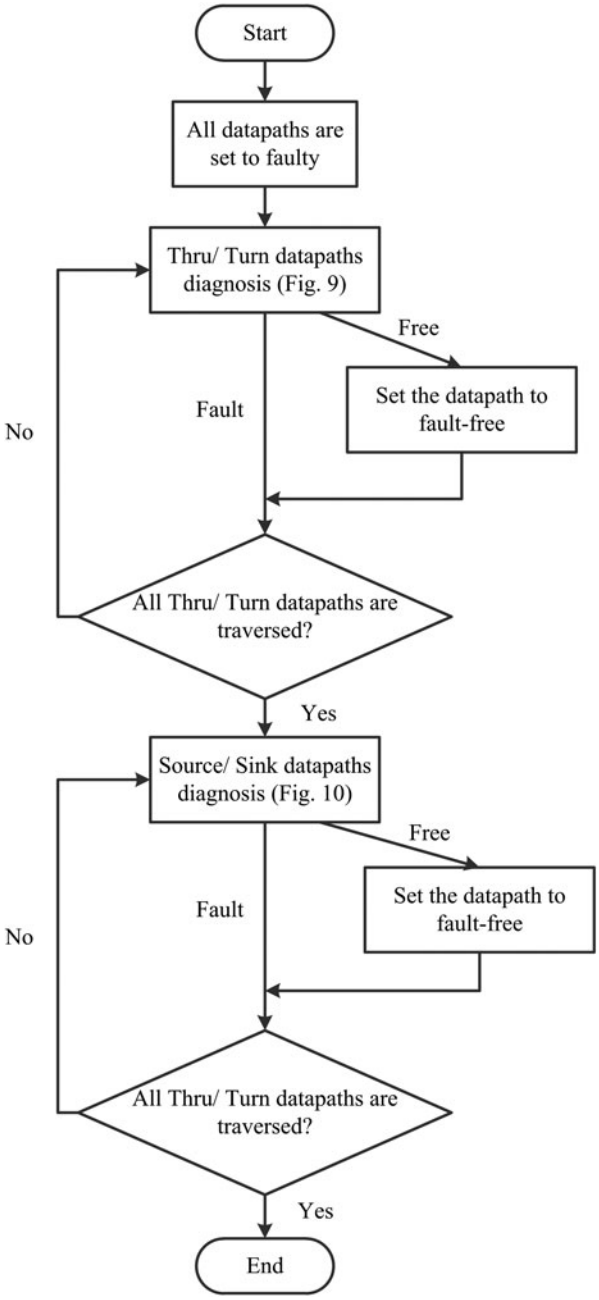

Fig. 7 The flow chart of diagnosis

network. Figure 8(a) shows an example to test the *Thru* and *Turn* datapaths. The test packet from the *TM A* to the *TM B* can test two *Thru* datapaths and one *Turn* datapath in the routers which are passed through. For an $n \times n$ *OCN*, $8n^2$ test packets are needed to test the whole *Thru* and *Turn* datapaths in the network.

Table 2 20 datapaths in the 20-path router model

Input Port	Output Port	Symbols
North	East, South, West, Local	*NE, NS, NW, NL*
East	North, South, West, Local	*EN, ES, EW, EL*
South	North, East, West, Local	*SN, SE, SW, SL*
West	North, East, South, Local	*WN, WE, WS, WL*
Local	North, East, South, West	*LN, LE, LS, LW*

Table 3 Three categories of the 20 datapaths in the 20-path router model

Categories	Datapaths
Turn	*NE, NW, EN, ES, SE, SW, WN, WS*
Thru	*NS, EW, SN, WE*
Source/Sink	*NL, EL, SL, WL, LN, LE, LS, LW*

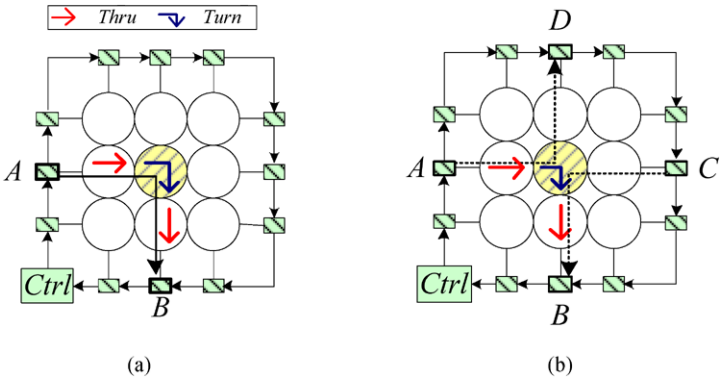

Fig. 8 (**a**) An example of the method to test two *Thru* and one *Turn* datapaths and (**b**) an example of the method to diagnose the *Turn* datapath

In order to reduce the test cycles, the paths between *TMs* can be tested in parallel. The routers in the diagonal lines can be tested simultaneously. The test packets from different *TMs* do not pass the same output port in a router simultaneously. Hence, no congestion occurs to increase the test cycles. Figure 9 represents the methods to test *Thru* and *Turn* datapaths with maximal parallelisms in a 3×3 *OCN*. There are three kinds of test diagonal lines, diagonal 1, diagonal 2, and diagonal 3. The routers in each diagonal line of Fig. 9 can be tested simultaneously. However, each mutual exclusive diagonal line, diagonal 1, diagonal 2, and diagonal 3, is tested sequentially.

The test packets can also diagnose the faulty datapaths. Figure 8(b) represents an example to diagnose the *Turn* datapath in Fig. 8(a). When *TM B* receives the faulty test packets from the *TM A*, it means that at least one of two *Thru* datapaths and one *Turn* datapath is faulty. If the *TM D* and the *TM B* can receive correct test packets from the *TM A* and the *TM C*, we can make sure that the *Turn* datapath is faulty. After checking all test packets in the test procedure with *Thru* and *Turn* datapaths diagnosis algorithm shown in Fig. 10, the faulty datapaths can be diagnosed.

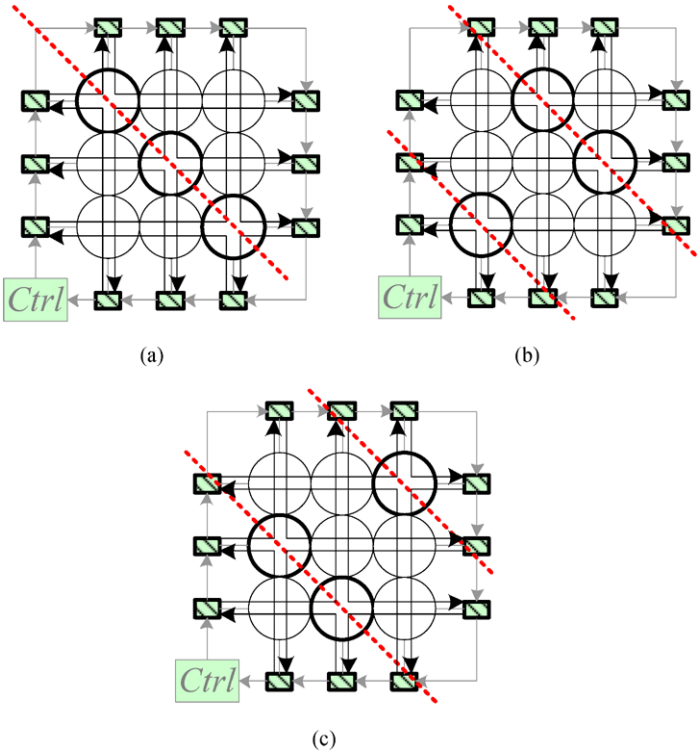

Fig. 9 Test *Thru* and *Turn* datapaths with maximal parallelism: (**a**) diagonal 1, (**b**) diagonal 2, and (**c**) diagonal 3

Fig. 10 The algorithm to diagnose the *Thru* and *Turn* datapaths

The diagnosis Algorithm for Thru and Turn datapaths

Initial of the Thru and Turn datapaths

All Thru and Turn datapaths are set to faulty

For each test packet ***p***

if ***p*** *is correctly transmitted between two TMs*

Set the Thru and Turn datapaths ***p*** *passed through to fault-free*

end

3.3.2 Test and diagnosis methods for Source/Sink datapaths

To test the *Source/Sink* datapaths, two *TMs* are selected in vertical or horizontal direction. Figure 11(a) represents an example to test the *Source/Sink* datapaths. The test packet from *TM A* to *TM C* can test two *Thru* and one *Source/Sink* datapaths. The router in each *NI* must add a simple circuit to change destination addresses and source addresses in test packets, which is pointed out in [8]. Each *TM* transmits test packets n times to another test module in an $n \times n$ mesh to test the *Source/Sink* datapaths in the router are passed through. $4n^2$ test packets are generated to the *Source/Sink* datapaths. The methods to test *Source/Sink* datap-

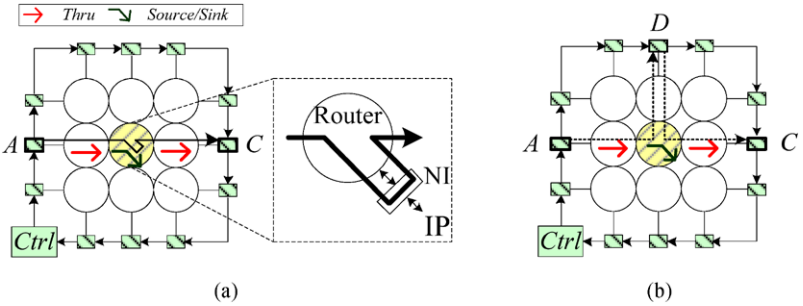

Fig. 11 (**a**) An example of the method to test two *Thru* and one *Source/Sink* datapaths and (**b**) an example of the method to diagnose the *Source/Sink* datapath

Fig. 12 Test *Source/Sink* datapaths with maximal parallelism: (**a**) vertical and (**b**) horizontal

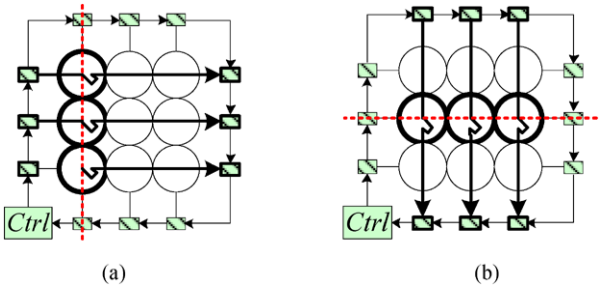

Fig. 13 The algorithm to diagnose the *Source/Sink* datapaths

The diagnosis Algorithm for Source/Sink datapaths

Initial of the Source/Sink datapaths

All Source/Sink datapaths are set to faulty

For each test packet ***p***

if ***p*** *is correctly transmitted between two TMs*

Set the Source/Sink datapaths ***p*** *passed through to fault-free*

end

aths with maximal parallelisms are also shown in Fig. 12. The router can be test vertically (in Fig. 12(a)) and horizontally (Fig. 12(b)).

The method to diagnose the *Source/Sink* datapaths is similar to the method of the *Thru* and *Turn* datapaths. Figure 11(b) represents an example to diagnose the *Source/Sink* datapaths in Fig. 12(a). When *TM C* receives the faulty test packets from the *TM A,* it means that at least one of two *Thru* datapaths and one *Sink* datapath is faulty. If the *TM D* and the *TM C* can receive correct test packets from the *TM A* and the *TM D*, the *Source/Sink* datapath is faulty. Figure 13 also represents the algorithm to diagnose the *Source/Sink* datapaths.

3.3.3 Fault-isolation methods

To isolate the impacts of the faulty components in the *OCNs* which are detected with the algorithms mentioned in Fig. 10 and Fig. 13, the *CTRL* generates the Fault-Isolation (*FI*)

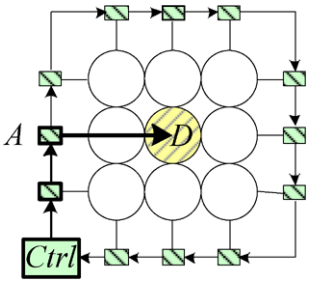

Fig. 14 The path from the *TM A* to the *TM D* to configure the faulty router (the router with oblique line)

packets to isolate the faulty components in the faulty routers. In the isolation method, we use the fault-free path in the test step to transmit the FI packets. Figure 14 represents a simple example. If the path from the *TM A* to the *Node D* is fault-free in the test and diagnosis processes, it can be applied to configure the faulty router (the router with oblique line). The *FI* packet must be transmitted from the *CTRL* to the *TM A* and then from the *TM A* to the *N*ode *D*.

3.4 Through-Path Fault-Tolerant Routing (*TP-FT*) algorithm

In order to recover the faulty *OCN* system, a fault-tolerant routing algorithm is necessary. Conventionally, the faulty router is fully closed while a fault is detected. However, faults may only influence certain functions of a faulty router, and undamaged parts can still work correctly if faulty parts are detected and located as shown in Fig. 1. Therefore, we can partially close the faulty path in the router and apply the fault-free datapaths in the faulty router.

To improve faulty network performance, this paper proposes a fault-tolerant routing algorithm, Through-Path Fault-Tolerant (*TP-FT*) routing algorithm. Using the 20-path router model shown in Table 2, we can only isolate the faulty path in the router rather than entire router. After knowing the faulty datapaths in the faulty router, one searching algorithm (as shown in Fig. 15) is needed to find the Through Paths (*TPs*) across the faulty router. The *TPs* are straight lines to avoid the deadlock problems in routings. The *SN*, *NS*, *EW*, and *WE* datapaths of the faulty routers are traced. According to the *SN*, *NS*, *EW*, and *WE* datapaths in the faulty routers, the *TPs* in the faulty *OCNs* can be defined. For instance, the *TP* in west-to-east direction is valid if the *WE* datapaths of the passing faulty routers are fault-free. *TPs* are applied for routings across the faulty router with shorter distance in comparisons with the traditional *FT* routings [11–13]. Figure 16 shows an example of four faulty routers in a 4×4 *OCN*. To trace the *TP* from S_0 to D_0, the *WE* datapaths of the faulty routers R_1 and R_3 must be checked. If the WE datapaths are fault-free, the TP from S_0 to D_0 is valid. The *TP* from S_1 to D_1 is also shown in Fig. 16. This TP is invalid because the SN datapath of the router R_0 is faulty, as the dotted arrow in Fig. 16. If there is no *TP* after doing *TP* search algorithm, some packets will become unreachable packets, because they cannot be transmitted to the destination. Three *TP-FT* routings, Through-Path Modified X-First routing (*TP-MXF*), Through-Path Extended XY routing (*TP-E-XY*), and Through-Path Chen and Chiu's routing (*TP-Chen and Chiu's*) are discussed in Sects. 3.4.1, 3.4.2, and 3.4.3, respectively.

3.4.1 Through-Path Modified X-First (TP-MXF) routing

The *TP-MXF* is extended by the *MXF* [11] by adding some *TPs* in the *MXF*. Figure 17(a) and (b) show two examples of extra *TPs* of *TP-MXF* for f-chains and f-rings, respectively. An

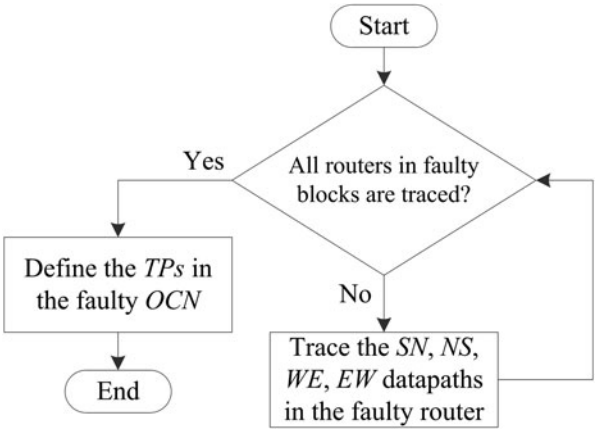

Fig. 15 The flow chart of the TP search algorithm

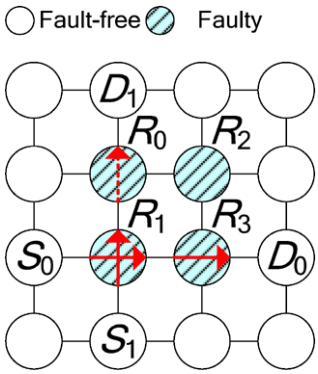

Fig. 16 An example to show the *TPs* from S_0 to D_0 and S_1 to D_1

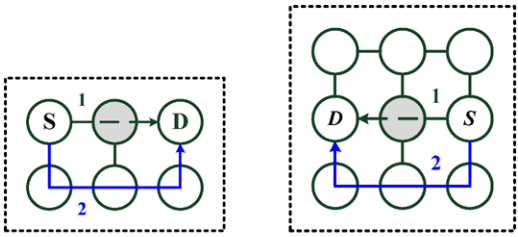

Fig. 17 The *TPs* of *TP-MXF* for (**a**) f-chains (**b**) f-rings

f-ring is a faulty region enclosed by a set of non-faulty routers; an f-chain is a faulty region located at the boundaries of mesh network [18]. Please note that the gray node represents faulty router; the node with "S" means source router; the node with "D" means destination router. The solid arrows are the routing paths in *MXF* and the dotted arrows represent the *TPs*. The numbers represent the priority of selecting routing paths in the routing process. If the router with dot is met in the routing process, the routing decision is decided. If the *TP* is valid, the packets pass through the *TPs*. If the *TP* is invalid, the packets follow the routing paths in *MXF*. *TP-MXF* uses the *TPs* to reduce the distances of routing paths and the traffics around the boundaries of the faulty routers.

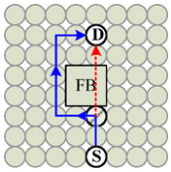

Fig. 18 An example of *TPs* of *TP-E-XY*

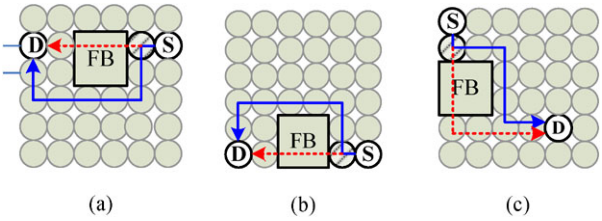

Fig. 19 Examples of *TP-Chen* and *Chiu's* to detour and (**a**) f-ring (**b**) f-chain (s-chain) (**c**) f-chain (non-s-chain)

3.4.2 Through-Path Extended XY (TP-E-XY) routing

The *TP-E-XY* is extended by the *E-XY* [12]. Figure 18 represents an example of *TPs* of *TP-E-XY*. The solid arrows are the routing paths in *E-XY* and the dotted arrows represent the *TPs*. *TP-E-XY* uses the *TPs* to reduce the distances of the routing paths and the traffics around the boundaries of the faulty blocks. Faulty block (*FB*) represents a block which contains a cluster of faulty routers.

3.4.3 Through-Path Chen and Chiu's (TP- Chen and Chiu's) routing

The *TP-Chen* and *Chiu's* is extended by the *Chen* and *Chiu's* [13]. *Chen* and *Chiu's* can support both f-chains and f-rings. Figure 19(a) shows the *TPs* of *TP-Chen* and *Chiu's* for the cases of a routing path to detour an f-ring in a 6×6 *OCN*. Figure 19 (b) and (c) show two examples of the *TPs* of *TP-Chen* and *Chiu's* for the cases an f-chain (s-chain and non-s-chain). The solid arrows are the routing paths in *Chen* and *Chiu's* routing and the dotted arrows represent the *TPs*. *TP-Chen* and *Chiu's* uses the *TPs* to reduce the distances of the routing paths and the traffics around the boundaries of the faulty routers.

4 A scalable *STR* architecture

According to the features of proposed *BISR* mechanism, we need to design a *STR* and fault-tolerant router architecture. In our design, we use the generic 5-port router without virtual channels as a design case to show the feature of our fault-tolerant circuit, and it is discussed in Sect. 4.1. Two fault isolation circuits must be inserted in fault-tolerant router described in Sect. 4.2. At last, the detail architecture of *STR* is described in Sect. 4.3.

4.1 Router architecture

Figure 20 illustrates the generic router architecture, which is modified from [19]. The router supports wormhole switching and round-robin scheduling algorithm. Virtual channels are restricted for low cost issues. The router contains five types of components: (1) five *FIFOs*, (2) five 4-to-1 *MUXs*, (3) five Address Decoders (*AD*s), (4) five Routing Logics (*RL*s), and (5) five 4×1 Arbiters (*ARB*s). The RLs and the ARBs are connected to the *FI*, which is discussed in Sect. 4.2.

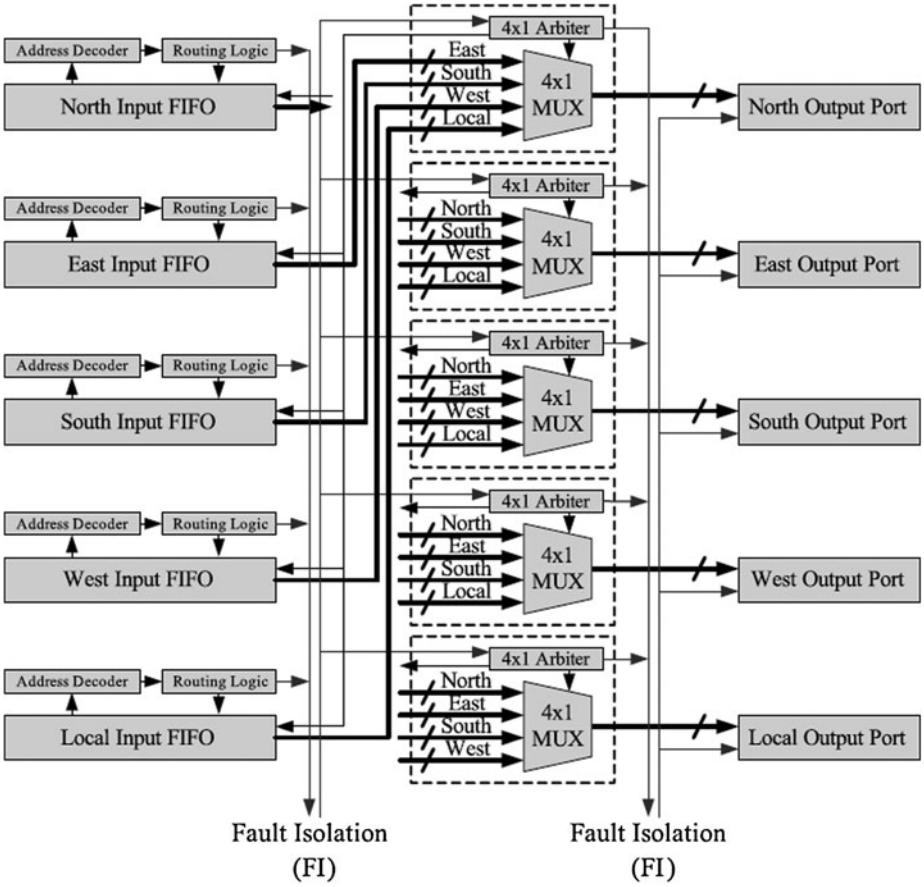

Fig. 20 The architecture of a generic router

4.2 Fault isolation circuit

The 20-path router can provide undamaged parts because the 20-path router contains the *FI* circuit to disable the influence of faulty *FIFOs* and *MUXs* in the generic wormhole router based on Table 4. Table 4 presents the impact of faulty path in 20-path router to the faulty components. After system recovery, the faulty components are masked by registers in Fig. 21(b) which are overwritten by recovery packets. Hence, packets can still be transmitted through undamaged parts of faulty 20-path routers. The *FI* circuit can be classified into two types: (1) Request-In Isolations (*RII*s) and (2) Request-Out Isolations (*ROI*s), which are described as below:

4.2.1 RIIs

The *RIIs* can reduce the influences of the faulty *FIFOs*. Figure 21(a) represents the block diagram of the *RII*s and the *ROI* in the North Output Port. For other ports, the cases are similar. The *Grant* signal from the *ARB* controls the selection of the 4-to-1 *MUX* from different input *FIFOs*. The inputs of the *RII*s are the *RL*'s output signals (*Erin*, *Srin*, *Wrin*,

Table 4 The Impacts of faulty *FIFOs* and *MUXs* in the *20PR*

Faulty Paths	Faulty Components
NE, NW, NS, NL	North input *FIFO*
EN, ES, EW, EL	East input *FIFO*
SN, SE, SW, SL	South input *FIFO*
WN, WE, WS, WL	West input *FIFO*
LN, LE, LS, LW	Local input *FIFO*
EN, SN, WN, LN	North output *MUX*
NE, SE, WE, LE	East output *MUX*
NS, ES, WS, LS	South output *MUX*
NW, EW, SW, LW	West output *MUX*
NL, EL, SL, WL	Local output *MUX*

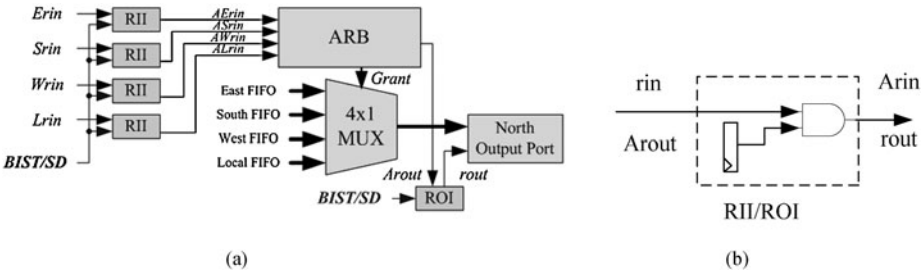

Fig. 21 The (**a**) block diagram and (**b**) architecture of the *RIIs* and the *ROI* in the north output port

and *Lrin* from East, South, West, and Local Input Port). If the isolation signal from the *BISR* identifies faulty *FIFOs*, the *RII* disables the input buffer request, *rin*. Hence, the *ARB* never selects the faulty *FIFOs* and the datapaths through faulty *FIFOs* are disabled. For example, if the East Input FIFO is faulty, *AErin* is disabled and no packets can pass through the *EN*, *ES*, *EW*, and *EL* datapaths.

4.2.2 ROIs

The *ROI*s can handle faulty *MUXs* in the 20-path router. Each *ROI* is connected to the *ARB* output signal (*Arout*) and the request output signal (*rout*) in the output port. If the isolation signal from the BISR identifies the faulty *MUX*, the ROI disables the request output signal, *rout*. Therefore, the datapaths through faulty *MUXs* are disabled. For example, if the *MUX* in Fig. 21(a) is faulty, the *rout* is disabled and no packets can pass through the *EN*, *SN*, *WN*, and *LN* datapaths.

4.3 Test module

STR contains two major components: (1) test module (*TM*) and (2) *CTRL*, as shown in Fig. 2. TMs generate and receive test packets through the connections between routers and test modules. The *CTRL* controls the operations of test modules and collects the test results by the connections between test modules and *CTRL*. The architecture of the test module is illustrated in Fig. 22. Each test module contains five blocks:

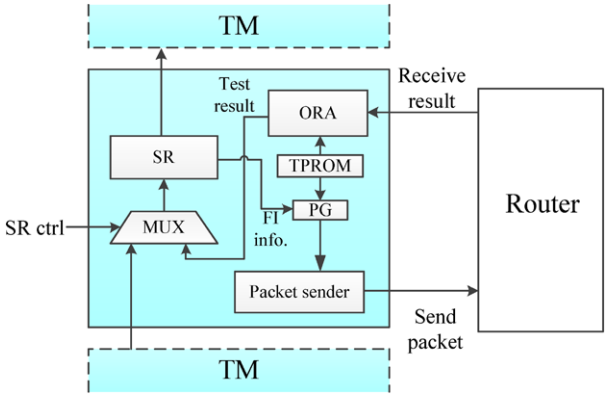

Fig. 22 The architecture of the *TM*

(1) Test Pattern ROM (*TPROM*)
The test patterns, which are design in *STR* architecture generation phase (shown in Fig. 3), are all stored in these ROMs.
(2) Packet Generator (*PG*)
It is charged of generating test packets (*FI* packets) with adding head, trail, and the test pattern sent from *TPROM* (*FI* information sent from controller).
(3) Packet sender
After packets generation, packet sender will send these packets to the router it attached to.
(4) Output response analyzer (*ORA*)
It receives test packets from other test modules and analyzes correctness of the packets.
(5) Shift Register (*SR*)
Test results of test packets are stored in *SRs* and transferred back to central controller by *SRs*. In addition, the controller can send *FI* information to appropriate test module through *SRs'* propagation.

CTRL is located at the corner of the *OCN* mesh, and connected to the head and tail of the chain formed by test modules. It controls the test flow of *STR* and analyzes the test results to identify faulty *FIFOs* and faulty *MUXs*.

The state diagram of the *CTRL* is illustrated in Fig. 23, as described as follows:

(1) The *CTRL* commands test module to send the next test packet until all test packets are sent (the order of test packets are implemented in *TM*s).
(2) *CTRL* commands shift registers in *TMs* to transfer test results back to *CTRL* through the ring. The test results are records of test packets' correctness.
(3) After all test results are received, *CTRL* performs diagnosis and identify faulty datapaths and components.
(4) The *CTRL* generates the FI packets and sent them to the router through the fault-free datapath.

5 Experiments and implementation

In this section, the system performance of generic OCN router test architecture (GR, the router is fully disabled which any parts of the router is faulty), *20PR* in [5], and surrounding

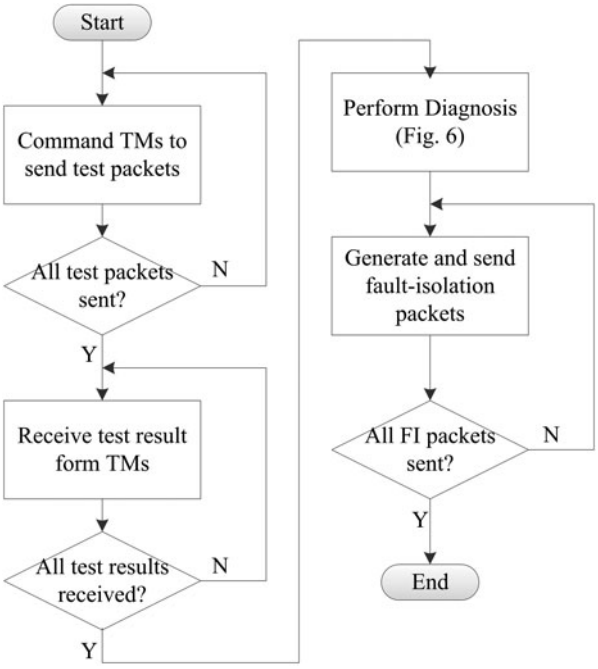

Fig. 23 The state diagram of the *CTRL*

test ring (*STR*) are compared. A SystemC simulator is developed for dynamic simulation of the network. In our simulation, 8×8 faulty meshes with 1, 2, and 4 faulty routers are generated. The networks route packets following proposed *TP-FT* routing algorithm in wormhole scheme, which takes one cycle for a flit to pass a hop, and each packet contains 5 flits. Furthermore, each faulty mesh is simulated under three types of synthetic traffic patterns: uniform random (*Rand*), exponential (*Exp*), and Rent's rule (*Rent*) [20].

5.1 Simulation of system performance

The more datapaths are disabled, the more packets are required rerouting. Therefore, the faulty system performance depends on currently used fault-tolerant routing algorithm. The ratio of unreachable packets is shown in Table 5. The unreachable packets reduced by 75.68∼83.29% and 68.33∼79.31% in *OCNs* using *20PR* and *STR*, respectively. We can observe that the ratio of unreachable packets in *OCN* using *20PR* and *STR* is lower than in *OCN* using GR, because GR fully disables this faulty router even only one fault is detected. *20PR* has slower unreachable packets since this approach has precise fault detection ability since embed *BISD* in each router.

In Table 6, the latencies in normal operation for GR, *20PR*, and *STR* under exponential and rent's traffic are compared. Since there are less rerouted packets for *20PR* and *STR*, the latency of GR is normalized as 1. *20PR* reduces latency by 5.49∼24.89%, and *STR* decreases latency by 4.86∼23.60% in different traffic patterns and number of faulty routers.

5.2 Test cycle

The timing of the *OCN* under test is described as followed: For every flit, if the *FIFO* is empty when the flit reaches the *FIFO*, it only takes one cycle to traverse through the router.

Table 5 Unreachable packets comparison of GR, *20PR*, *STR*

Traffic	Approach	Number of faulty routers					
		1		2		4	
		Unreachable Ratio	Reduction	Unreachable Ratio	Reduction	Unreachable Ratio	Reduction
Rand	GR	11.29%	–	19.23%	–	34.27%	–
	20PR	2.75%	75.68%	4.31%	77.59%	7.66%	77.65%
	STR	3.31%	70.69%	5.45%	71.67%	8.70%	74.59%
Exp	GR	3.35%	–	6.29%	–	11.96%	–
	20PR	0.63%	81.12%	1.39%	77.86%	2.01%	83.20%
	STR	0.97%	71.03%	1.99%	68.33%	2.55%	78.65%
Rent	GR	3.19%	–	5.93%	–	11.25%	–
	20PR	0.60%	81.19%	1.34%	77.46%	1.88%	83.29%
	STR	0.86%	73.11%	1.81%	69.56%	2.33%	79.31%

Table 6 Latency comparison of GR, *20PR* and *STR*

Traffic	Approach	Fault-free	Number of faulty routers					
			1		2		4	
			Latency	Reduction	Latency	Reduction	Latency	Reduction
Rand	GR	14.79	20.08	–	19.69	–	20.01	–
	20PR		17.99	10.42%	16.78	14.74%	15.20	24.02%
	STR		18.30	8.86%	16.87	14.31%	15.28	23.60%
Exp	GR	16.31	24.45	–	24.94	–	26.97	–
	20PR		22.67	7.27%	21.83	12.48%	20.26	24.89%
	STR		23.10	5.49%	22.44	10.02%	20.79	22.93%
Rent	GR	17.55	26.22	–	26.84	–	29.20	–
	20PR		24.32	7.25%	23.48	12.51%	22.03	24.57%
	STR		24.94	4.86%	23.86	11.12%	22.44	23.16%

No buffer unit is placed in the output channel. Table 7 represents the comparison of different testing methods in 4×4 mesh. It represents that the test cycle of proposed *STR* is less than the approaches in [3, 6, 7, 10]. Because *STR* approach supports parallel testing, the test cycle is less than the DfT-based approaches in [6, 7, 10]. Because our fault model, which described in Sect. 3.1, only considers the faults in *FIFO's D*-flip-flop and crossbar (implemented in *MUX*), the faults in *FIFO* access logic and interconnection do not be considered in fault coverage estimation. In addition, *20PR* in [5] uses the same fault model. Therefore the fault coverage of *20PR* is the same as *STR*.

5.3 Area overhead

Table 8 represents each component area in a 20-path router. We can observe that the area of *FIFO* and *MUX* occupy the most area in a router. This is the reason that we only consider the fault model in *FIFOs* and *MUXs* in this paper. The overheads of STR with respect to different mesh size are shown in Table 9. Because number of test modules grow in $O(n)$ while overall

Table 7 Comparison of *OCN* test architecture on 4 × 4 mesh

	Fault Coverage	Area Overhead	Test Cycle
Hosseinbady *et al.* [6]	95.20%	6.95%	4.05×10^5
Amory *et al.* [7]	98.93%	9.17%	$9.45 \times 10^3 \sim 3.33 \times 10^4$
Peterson *et al.* [3]	99.89%	N/A	2.74×10^3
Raik *et al.* [10]	99.81%	N/A	2.24×10^2
20PR [5]	97.79%	15.17%	1.17×10^2
Proposed *STR*	97.79%	10.90%	1.94×10^2

Table 8 Router component area of *STR*

Components	Area (μm^2)	Area Ratio
FIFO	31,205	76.12%
MUX	5,135	12.53%
RL, ARB	3,636	8.87%
Redirector	874	2.13%
Others	145	0.35%
Total	40,995	100%

Table 9 Area overhead of components in *STR*

	3 × 3	4 × 4	5 × 5	6 × 6
Router with FI	368,955 μm^2	655,920 μm^2	1,024,876 μm^2	1,475,820 μm^2
TM	33,408 μm^2	44,544 μm^2	55,680 μm^2	66,816 μm^2
CTRL	1,375 μm^2	1,832 μm^2	2,299 μm^2	2,755 μm^2
Others	113 μm^2	152 μm^2	181 μm^2	221 μm^2
Total	403,851 μm^2 (+13.3%)	702,448 μm^2 (+10.9%)	1,083,036 μm^2 (+9.45%)	1,545,612 μm^2 (+8.48%)

area of routers grows in $O(n^2)$, the extra overhead for STR decreases in percentage as mesh size grows. The architecture is implemented by Verilog RTL and synthesized with TSMC 0.13 µm technology. The generic $X-Y$ router is an OCN router with 34-bit data width and 4-flit input buffer in each input channel and no virtual channel is used. Comparing to *OCNs* using generic $X-Y$ router, the area overhead of *OCNs* with *STR* is less than 13.3%. Table 7 also represents the area overhead in comparison with some related researches in 4 × 4 mesh. The approaches in [5] embedded one *BISD* in each router, the area overhead is the heaviest. Because DfT-based approaches in [6] and [7] do not need test pattern generators or output response analyzers hardware, the area overhead is lighter than *STR*.

6 Conclusions

This paper proposes a scalable *BISR* design methodology to detect and recover faults during warm-up time and maximize routing paths with TP-FT fault-tolerant routing algorithm

when fault happens. The proposed *STR* VLSI architecture for *BISR* consists of test modules to extend self-diagnosis of previous works to self-recovery and fault-isolated router architecture for fault recovery. The test modules are wrapped to *OCN* in ring topology. In our experiments, comparing to previous solutions, which disable the entire faulty router, *TP-FT* routing algorithm reduces 68.33∼79.31% unreachable packets and 4.86∼23.6% latency. Besides, area overhead of *STR* is less than 13.3%, and the test cycle is less than most of other related works.

Acknowledgements This work was supported by the National Science Council of Taiwan under Grant NSC 98-2220-E-002-034 and NSC 97-2221-E-002-239-MY3.

References

1. Benini L, Micheli GD (2002) Network on chip: a new paradigm for systems on chip design. In: IEEE proceedings of the conference on design, automation and test in Europe conference and exhibition, pp 418–419
2. Semiconductor Association (2005) The international technology roadmap for semiconductor (ITRS)
3. Petersen K, Oberg J (2007) Toward a scalable test methodology for 2D mesh network-on-chips. In: Proceedings of the conference on design, automation and test in Europe (DATE '07), pp 367–372
4. Grecu C, Pande P, Ivanov A, Saleh R (2006) BIST for network-onchip interconnect infrastructures. In: Proceedings of 24th IEEE VLSI test symposium, April, 2006, pp 30–35
5. Lin S-Y, Shen W-C, Hsu C-C, Chao C-H, Wu A-Y (2009) Fault-tolerant router with built-in self-test/self-diagnosis and fault-isolation circuits for 2D-mesh based chip multiprocessor systems. In: Proc IEEE int symp VLSI design, automation, and test (VLSI-DAT-2009), pp 72–75
6. Hosseinabady M, Banaiyan A, Bojnordi MN, Navabi Z (2006) A concurrent testing method for NoC switches. In: Proceedings of the conference on design, automation and test in Europe (DATE '06), Munich, Germany, pp 1171–1176
7. Amory AM, Briao E, Cota E, Lubaszewski M, Moraes FG (2005) A scalable test strategy for network-on-chip routers. In: Proceedings of IEEE international test conference (ITC '05), Nov 2005, pp 591–599
8. Raik J, Govind V, Ubar R (2006) An external test approach for network-on-a-chip switches. In: 15th Asian test symposium (ATS'06), pp 437–442
9. Raik J, Ubar R, Govind V (2007) Test configuration for diagnosing faulty links in NoC switches. In: 12th IEEE European test symp (ETS'2007), pp 29–34
10. Raik J, Govind V, Ubar R (2009) Design-for-testability-based external test and diagnosis of mesh-like network-on-a-chips. IET Comput Digit Tech 3:476–486
11. Zhang Z, Greiner A, Taktak S (2008) A reconfigurable routing algorithm for a fault-tolerant 2D-mesh network-on-chip. In: 45th ACM/IEEE design automation conference
12. Wu J (2003) A fault-tolerant and deadlock-free routing protocol in 2D meshes based on odd-even turn model. IEEE Trans Comput 52:1154–1169
13. Chen K-H, Chiu G-M (1998) Fault-tolerant routing algorithm for meshes without using virtual channels. J Comput Inf Sci Eng 14:765–783
14. Yuan F, Huang L, Xu Q (2008) Re-examining the use of network-on-chip as test access mechanism. In: Design, automation and test in Europe (DATE '08), Mar 2008, pp 808–811
15. Amory AM, Goossens K, Marinissen EJ, Lubaszewski M (2006) Wrapper design for the reuse of networks-on-chip as test access mechanism. In: Proceedings of the eleventh IEEE European test symposium, pp 213–218
16. Cota E, Liu C (2006) Constraint-driven test scheduling for NoC-based systems. IEEE Trans Comput-Aided Des Integr Circuits Syst 25(11):2465–2478
17. Wang L-T, Wu C-W, Wen X (2006) VLSI test principles and architectures. Morgan Kaufmann, San Francisco
18. Boppans RV, Chalasani S (1995) Fault-tolerant wormhole routing algorithms for mesh networks. IEEE Trans Comput 44:848–864
19. Hu J, Marculescu R (2004) Application-specific buffer space allocation for networks-on-chip router design. In: IEEE/ACM international conference on computer aided design (ICCAD-2004), Nov 2004, pp 354–361
20. Greenfield D et al (2007) Implications of Rent's rule for NoC design and its fault-tolerance. In: First international symposium on networks-on-chip, pp 283–294