Microprocessors and Microsystems 51 (2017) 252–263

Contents lists available at ScienceDirect

Microprocessors and Microsystems

journal homepage: www.elsevier.com/locate/micpro

Minimizing the system impact of router faults by means of reconfiguration and adaptive routing

Junshi Wang[a,b,*], Masoumeh Ebrahimi[c], Letian Huang[a,b], Qiang Li[b], Guangjun Li[b], Axel Jantsch[d]

[a] Technische Universität Wien, Vienna, Austria
[b] University of Electronics Science and Technology of China, Chengdu, Sichuan, PR China
[c] Royal Institute of Technology, Sweden and University of Turku, Finland
[d] Institute of Computer Technology, Technische Universität Wien, Vienna, Austria

ARTICLE INFO

Article history:
Received 3 July 2016
Revised 6 February 2017
Accepted 14 February 2017
Available online 26 April 2017

Keywords:
Reconfigurable router architecture
Bypassing channels
Adaptive routing algorithm

ABSTRACT

To tolerate faults in Networks-on-Chip (NoC), routers are often disconnected from the NoC, which affects the system integrity. This is because cores connected to the disabled routers cannot be accessed from the network, resulting in loss of function and performance. We propose *E-Rescuer*, a technique offering a reconfigurable router architecture and a fault-tolerant routing algorithm. By taking advantage of bypassing channels, the reconfigurable router architecture maintains the connection between the cores and the network regardless of the router status. The routing algorithm allows the core to access the network when the local router is disabled.

Our analysis and experiments show that the proposed technique provides 100% packet delivery in 100%, 92.56%, and 83.25% of patterns when 1, 2 and 3 routers are faulty, respectively. Moreover, the throughput increases up to 80%, 46% and 33% in comparison with FTLR, HiPFaR, and CoreRescuer, respectively.

1. Introduction

To detect and tolerate faults on the control paths of the routers, it is very common to isolate routers from the running NoC temporarily or permanently. However, it disturbs the integrity of the system and is not always necessary. Routers might be disabled for different reasons. In [1], the router under test (RUT) is disabled from the network for on-line testing, and thus packets have to be blocked in the neighboring routers until the testing procedure is completed. As another example, the router with faulty units (such as arbiters) may be abandoned if no other method has been applied to tolerate faults [2,3]. Every router has the probability to be faulty or abandoned, and all of them should be tested as well. RUTs and abandoned routers may block the communication between the core and the rest of the network. Moreover, the entire system may be stopped due to data dependency and coherency.

NoCs provide inherent path redundancy and a fault-tolerant infrastructure [4]. Fault-tolerant routing is an effective solution to tolerate disabled links and routers by reconfiguring paths. Different techniques are exploited to equip NoCs with fault tolerance [2]. Fault-tolerant routing algorithms including Dynamic Routing Protocol for NoCs (DR-NoC) [5], Fault-on-Neighbor (FoN) [6], Q-Learning based routing [7], Simple Flooding Algorithm, Directed Flooding Algorithm, and Redundant Random Walk Algorithm [8] can be executed in NoCs to bypass the disabled areas. The basic assumption behind these methods is that the core is disabled if the local router is disabled. However, this is not always necessary and may have a severe impact on the functionality of the entire system or its performance.

In our previous work [9], we introduced a routing algorithm, *CoreRescuer*, which keeps alive the core connected to a disabled router (either because of faults or testing the router). The disabled router is reconfigured to act as a bypassing router, maintaining the connectivity between the routers in the horizontal and vertical directions. The main features of the CoreRescuer algorithm are as follows:

1. Cores continue to be connected to the network when their local routers are disabled.

* Corresponding author.
E-mail addresses: wangjsh@std.uestc.edu.cn (J. Wang), masebr@utu.fi (M. Ebrahimi), huanglt@uestc.edu.cn (L. Huang), qli@uestc.edu.cn (Q. Li), gjli@uestc.edu.cn (G. Li), axel.jantsch@tuwien.ac.at (A. Jantsch).

http://dx.doi.org/10.1016/j.micpro.2017.02.004
0141-9331/

2. An adaptive fault-tolerant routing algorithm is utilized which supports the access of the cores through bypassing channels.
3. CoreRescuer can tolerate disabled routers. Also, the faulty units in routers can be bypassed. However, faults on links are not covered.

In this paper, we propose an enhanced mechanism for the mesh topology, called *E-Rescuer* advancing the features of CoreRescuer as follows:

1. The effect of the disabled router has been limited to a 3 × 3 area whereas a *fully* adaptive routing algorithm is applied to the rest of the network. Note that in this context *fully* means all available minimal paths. The proposed idea results in the performance improvement of up to 33% over CoreRescuer. This observation highlights the importance of adaptivity in maintaining the network performance when faults are presented. The paper includes a complete Pseudo-code, showing the simplicity of the algorithm.
2. Our analysis provides precise reliability values. The obtained values are compared with experimental results, leading us to an observation that the commonly used traffic pattern, uniform traffic, overestimates reliability values as it does not necessarily identify all possible deadlock scenarios.

Also, we employ a bypassing channel, introduced in [9], and demonstrate that it improves performance and reduces hop count because the disabled router offers shortcut channels.

Moreover, we have visualized the traffic distribution of E-Rescuer and CoreRescuer to emphasize the fact that E-Rescuer achieves a better traffic balance by distributing packets to peripheral regions.

Faults in a router can be classified into control-path and data-path faults. E-Rescuer focuses on the control path failures of routers because the control path failures could cause the time-slot chaos and even deadlocks which cannot be easily recovered (e.g. one solution is reset). On the other hand, faults on the data paths can be tolerated by commonly used methods such as Error-Correcting Codes (ECC) or retransmission. E-Rescuer may not be suitable for transient faults as these types of faults last for few cycles and it is not worth to reconfigure the router architecture.

The paper is organized as follows. After a review of related work, the reconfigurable router architecture is described in Section 3, and the E-Rescuer routing algorithm is described in Section 4. The simulation setup and the experimental results are reported in Section 5 while the summary and conclusion are given in the final section.

2. Related work

Fault-tolerant routing algorithms [2] in NoCs can be classified into different categories as tolerating faults in links [10–16], routers [17–21], or both [6,7,22].

The main concerns in fault-tolerant routing algorithms are maintaining network integrity and keeping the network free from deadlock when faults are introduced. To satisfy these goals, table-based methods have received popularity where tables can be updated after the occurrence of faults [7,10–12]. Table-based methods can be differentiated by techniques to fill out tables and capability to find deadlock-free routes. For example in [7], reinforcement learning is applied to find all fault-free routes while in [12] the fault-aware routing algorithm is enlightened by the concept of the ant colony optimization. The table-based routing algorithms impose an area overhead of routing tables.

On the other hand, reconfigurable router architectures and routing algorithms have been proposed as light-weight solutions where the mentioned goals could be obtained [6,13,14,17,23]. Among them, LBDR [23] relies on a combinatorial logic circuit to compute the output port for each hop. In [17], reconfiguration is made to route packets through a cycle free contour surrounding a faulty router by knowing the faulty status of eight direct and indirect neighboring routers. The presented algorithm in [6] is able to tolerate both faulty links and routers but at the cost of complexity and hardware overhead.

The focus of the mentioned methods is on improving reliability than performance and throughput. One feasible approach to enhance performance is to increase routing adaptivity. In this direction, the HiPFaR routing algorithm [20] tries to improve reliability and performance at the same time. This goal has been achieved by utilizing an adaptive routing algorithm and selecting a fault-free path prior reaching faults. Another adaptive and fault-tolerant routing is presented in [16] which is based on the selection of an intermediate node for each source and destination pair. Packets are routed to their destinations via the intermediate node. Although these methods can reduce latency but still the performance gap is considerable.

All the above fault-tolerant algorithms are built based on the assumption that cores connected with the faulty routers are abandoned and disconnected from the network. However, we argue that this assumption is not always realistic and may affect system functionality or performance. In [18], a bus shared by all input and output ports is utilized to replace the crossbar unit. The cores can access the network through the bus. However, the buffers become critical components after reconfiguration. In a technique called NoRD [19] all cores are connected through a ring by a separate virtual channel so that the cores connected to powered-off routers can still access the rest of the network. The main shortcoming of NoRD lies on its performance loss in large networks as the ring becomes very long.

The use of multiple network interfaces is another means to keep cores connected when routers fail. In [24], two topologies with 2 and 4 network interfaces for one core are investigated. One core is not isolated from the network unless all connected routers are disabled. The disadvantage of this approach is the hardware overhead of the network interface (NI) and ports.

CoreRescuer [9] suggests a complete solution to keep alive the core connected to a disabled router. It is composed of a reconfigurable router architecture with bypassing channels and a fault-tolerant routing algorithm based on minimal paths. CoreRescuer is able to tolerate faults while maintaining integrity and performance. However, the routing rules disturb the traffic balance in the network and may lead to hotspot areas which are also common to other fault-tolerant methods. In other words, although CoreRescuer employs an adaptive routing and satisfies the connectivity of all cores to the network, congestion resulting in decreased performance remains an issue.

In this paper, we introduce a method that advances the current state-of-the-art by satisfying the main features of performance, throughput, reliability, integrity, and core connectivity through a reconfigurable router architecture and routing algorithm. The routing algorithm offers full adaptivity as long as packets do not reach the 3 × 3 area, centered by a faulty router. This characteristic maintains performance and throughput level when faults are presented. Network integrity is obtained by reconfiguring a faulty router as a bypassing router and connecting routers in horizontal and vertical directions. Through these bypassing channels, all cores remain connected to the network despite the faulty routers. The algorithm offers high reliability which has been calculated through both theoretical analysis and experimental evaluation.

Table 1
Default bypassing connections.

Input channels	L	E	W	N1	N2	S1	S2
Output channels	N2	W	E	S1	L	N1	S2
Output channels of top borderline routers	S1	W	E	–	–	L	S2

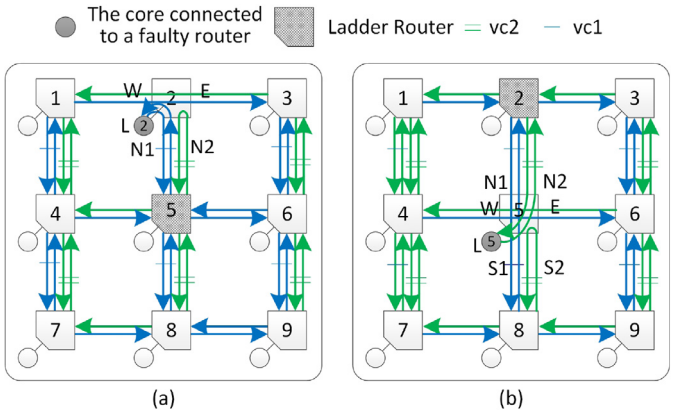

Fig. 1. The use of bypassing channels is illustrated when the disabled router (a) is located in the top row or (b) when it is NOT located in the top row.

3. The router architecture of E-Rescuer

3.1. Default router architecture

The default router architecture is similar to but different from the CoreRescuer architecture [9]. The difference is on the way that default bypassing connections, e.g. in Table 1, is planned to achieve the goal of the algorithm. Each router has seven pairs of channels, i.e. Local (L), East (E), West (W), North1 (N1), North2 (N2), South1 (S1), and South2 (S2). By default, the input and output channels are connected through a crossbar unit. E-Rescuer has some default paths such that the input channels connect to the output channels directly when the router is disabled. Thereby, in the case of faults, the bypassing channels connect the input and output ports in a static manner. Table 1 lists the static connections for the top borderline routers and the rest of the routers. As can be seen from this table, the local core can receive packets from the input ports S1 or N2 depending on whether the disabled router is located in the top borderline or not. The core can also deliver packets to the network by the static connection between the input port L and the output ports S1 (i.e. for top borderline routers) or N2 (i.e. for all the other routers) ports.

The north or south neighbor of the disabled router, which helps the packets to reach the core of the disabled router, is called *ladder router*. The bypassing connections are not only beneficial to keep the integrity of the network but also useful for reducing the latency.

Fig. 1(a) shows the bypassing connections in a 3 × 3 network where the router 2, located in the top borderline, is disabled, and the router 5 acts as a ladder router. In Fig. 1(b), reversely, the router 5 is disabled, and the router 2 acts as a ladder router.

Multiplexers (Mux) and de-multiplexers (Demux) are implied to switch between the bypassing channel and the original data path in the router. The reliability is not only related to the area but also to temperature, voltage, utilization, etc. Higher utilization contributes to higher failure rates [25]. Since the Mux and Demux share the same utilization and environment (temperature and voltage), they do not suffer more errors than the other parts. Therefore, the relative area determines the relative failure rates. According to the area report from Design Compiler, the Mux and Demux only take 2.54% of the router area (As shown in Table 4), implying that the fault probability of Mux and Demux is low. To improve the reliability of Mux and Demux, other circuit and physical design methods such as larger feature size of gates should be used, which is beyond the scope of this paper.

The reconfiguration of bypass channels is applied after the fault detection. We assume that an on-line Build-in Self-Test (BIST) fault detection mechanism (e.g. [1]) is employed at each router to detect and diagnose the errors. The BIST mechanism will trigger the reconfiguration if the errors on the control path are detected. The testing wrapper of BIST always contains the entire router as shown in [1]. During the testing procedure of BIST, packets are blocked in the neighboring routers to ensure that no packet remains in the router under test (RUT). After BIST, the RUT is reconfigured to either bypass channels if faults are detected, or the normal data paths if faults are recovered. After the reconfiguration, the algorithm should be able to deliver all packets to their destinations. Thus, no packet is dropped.

3.2. Rules for deadlock and livelock freedom

To prove the freedom from deadlock, we virtually partition the network into two completely disjoint subnetworks. These two subnetworks cover disjoint sets of channels as subnetwork A {E, N1, S1} and subnetwork B {W, N2, S2}.

Since each of the subnetwork A and subnetwork B covers three directions as (E, N1, and S1) and (W, N2, and S2) respectively, there is no possibility to form a complete cycle within each subnetwork. Also, because subnetworks are disjoint from each other, a cycle cannot be formed between the subnetworks. As a result, the network is deadlock free if packets are solely belonging to either the subnetwork A or the subnetwork B. To improve the routing flexibility but still avoid deadlock, packets in the subnetwork B can safely switch to the subnetwork A but not vice versa. No circle can form between the subnetworks avoiding the possibility of deadlock.

4. The E-Rescuer routing algorithm

As we discussed, traditional fault-tolerant routing algorithms suffer from disconnecting the healthy functional cores when their routers are disabled. E-Rescuer addresses this issue with bypassing channels. Another important shortcoming is that a disabled router influences the movement of all packets in the network, leading to a significant performance loss. E-Rescuer limits the packets' movement inside a 3-by-3 node area. Such limitations result in a performance improvement of up to 33% because a fully adaptive routing is still applied to all packets outside of this area under the presence of faults. In summary, E-Rescuer helps in a better distribution of packets all over the network.

In this section, we first describe the propagation of the router status (Disabled or Enabled) in the 3-by-3 area. Then the basic form of the E-Rescuer algorithm without considering the 3-by-3 area is explained. After that, we investigate the situations inside the 3-by-3 area. Finally, the entire algorithm is given in Pseudo code.

4.1. Router status propagation

By default, all the routers are assumed to be enabled. With the occurrence of a fault, as shown in Fig. 2, the direct and indirect neighboring routers are informed about the status of the disabled router. For this purpose, each router transmits its router status to the four neighboring routers. In addition, the router status of each neighboring router is transferred to the next neighboring router in the clockwise direction. For example, the status of the eastern neighbor is transferred to the north.

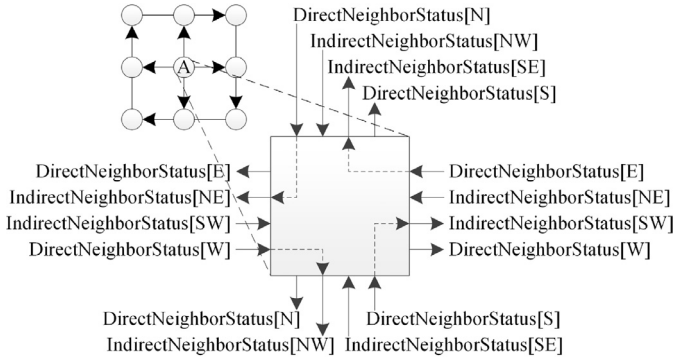

Fig. 2. The mechanism of the router status propagation.

The router status information is propagated using 1-bit signals, described as follows:

1. *DirectNeighborStatus*: Sending the enable/disable status of the local router to the neighboring router (output) or vice versa (input);
2. *IndirectNeighborStatus*: Transferring the enable/disable status of the neighboring router to the next neighboring router in the clockwise direction.

Because the status signals are identified by their related location, only one bit presenting the status is necessary. The status propagation is used to recognize the 3-by-3 areas. These regions are called *distinct regions* as different routing rules are applied inside them.

The aim of the distinct region is to limit the effect of disabled routers. Packets can choose their direction freely outside this region and do not need to consider the related location with the disabled routers. Therefore, the distinct region should be large enough so that the packets can be routed from all routers to a ladder router or vice versa. The 3-by-3 region is the minimal area to meet this condition.

4.2. Path choices outside the distinct regions

Fully adaptive routing is used to route packets outside of the distinct region. The NW and SW packets can take the W, N2, and S2 channels without any limitation (Fig. 3(a)) while The NE and SE packets can use any of the E, N1, and S1 channels freely (Fig. 3(b)). The selection between two possible directions is made based on the congestion values (i.e. the free buffer slots in the neighboring routers). For example, as shown in Fig. 3(a), at each router, NW packet can be routed through the less congested direction by choosing from the N or W direction.

Routers receive the congestion status of the neighboring routers using a credit-based flow control mechanism. The congestion status is the number of free slots in the corresponding input buffer of the neighboring router. Among the available directions, a packet is sent to the direction with more free slots.

4.3. Path choices inside the distinct regions

In this subsection, different routing paths inside the distinct region are investigated.

4.3.1. The source router is disabled

Fig. 4(a) indicates the routing paths to show that packets can reach any core in the network if delivered from a source core connected to a disabled router. As shown in this figure, packets are

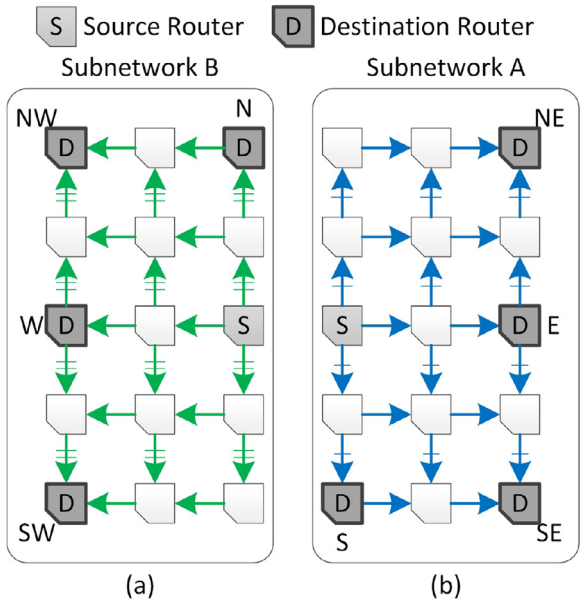

Fig. 3. The path choices of the E-Rescuer algorithm outside the distinct region. Packets can travel from the source router to the destination routers along several paths shown by the arrows. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

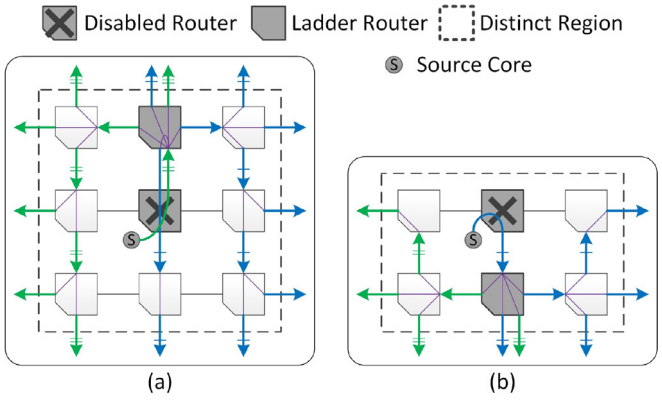

Fig. 4. The path choices inside the distinct region when the source router is disabled.

delivered to the ladder router by using the N2 channel first. Depending on the destination position, packets switch to the subnetwork A (i.e. for E, S, NE, and SE-bound packets) or continue routing in the subnetwork B (i.e. for W, N, NW, and SW-bound packets). Thereby, all packets can reach from any source core to any destination core even though the source router is disabled. In the ladder router, the south channel is used only for southward packets. When the disabled source router is located in the top borderline, Fig. 4(b), packets are first sent to the south neighboring router using the S1 channel and then delivered toward any other cores in the network. As the N1 port of the ladder router is directly connected with the rescued core, the turn from the N1 port to S2 port (Fig. 4(b)) does not lead to a deadlock risk.

4.3.2. The destination router is disabled

Fig. 5(a) shows the routing options when packets are sent to the destination core that is connected to a disabled router. Packets should be sent to the ladder router in order to reach the desired core. For this purpose, packets entering the distinct region through the southern part (i.e. SE, SW, and S) are first sent to the same column as the destination core. In this column, packets are routed

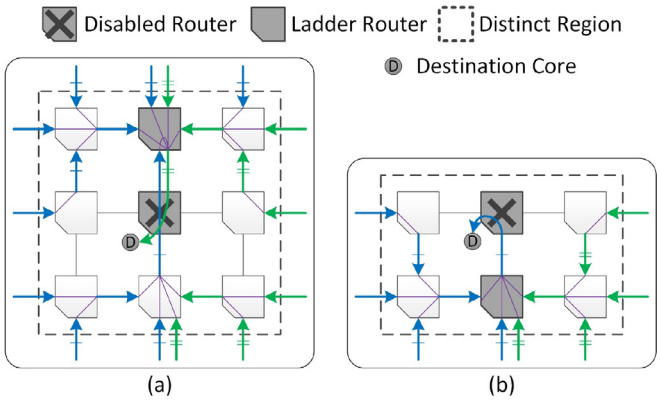

Fig. 5. The path choices inside the distinct region when the destination router is disabled.

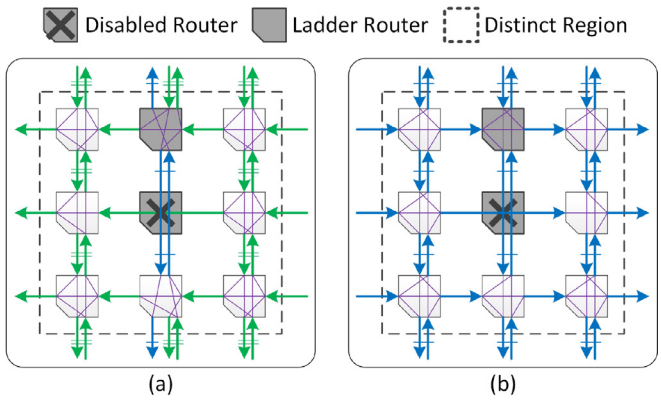

Fig. 6. The path choices inside the distinct region when neither the source nor the destination is disabled.

toward the ladder router using the N1 channel. The ladder router is connected to the destination core by the S2 channel. Packets entering the distinct region from the other directions (i.e. W, E, N, NE, NW) are directly routed to the ladder router. If the disabled router is in the top borderline (Fig. 5(b)), packets are delivered to the ladder router, i.e. located in the south neighboring router, and then to the destination core using the N1 channel. It is obvious that all packets can reach from any source core to any destination core even though the destination router is disabled. As the S2 port of the ladder router is directly connected with the rescued core, the turns from the S1 and N1 port to S2 port (Fig. 5(a)) do not lead to a deadlock risk.

4.3.3. The disabled router is neither source nor destination

If none of the source and the destination routers are disabled, a partially adaptive routing is utilized for routing packets inside the distinct region. Fig. 6(a) shows the available paths for W, NW, SW and N-bound packets while Fig. 6(b) shows the available paths for E, NE, SE and S-bound packets, respectively. In Fig. 6(b) all packets can use the bypassing channels without limitations as all channels belong to subnetwork B. However, in Fig. 6(a), only when the destination is located in the same column as the disabled router, packets can use the bypassing channel in the vertical direction and continue routing toward the destination by remaining in the subnetwork A. This is to assure that packets do not switch between subnetworks twice.

The E-Rescuer algorithm respects the general rules of channel assignments in all conditions without exception, which guarantees that the algorithm is deadlock and livelock free. In other words, in all cases, packets are either belonging to the subnetwork A or the subnetwork B. In addition, packets belonging to the subnetwork A can switch to the subnetwork B but not vice versa.

Table 2
The required number of hops for eight positions of source and destination cores (x is the Manhattan distance between source and destination).

Destination-Source Position					
Disabled Router's Position	E/W	N	S	NE/NW	SE/SW
Not on Path	x	x	x	x	x
Intermediate	x−1	x−1 a: x−1	x−1	x−1 a&e: x−1	x−1 c: x+1
Destination	x+1	b: x	x−1	b&e: x c: x+1	d: x−1
Source	x+1	x−1	a: x−1 b: x	x−1	a: x−1 b: x+1

The conditions leading to the given number of hops: [a] The disabled router is in the top borderline. [b] The disabled router is not in the top borderline. [c] Packet enters the distinct region through the E or W neighbor of the disabled router. [d] Packet enters the distinct region through the NE, NW and N neighbor of the disabled router. [e] Packet enters the distinct region through the SE, SW and S neighbor of the disabled router.

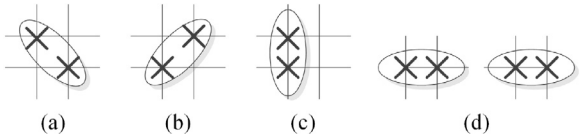

Fig. 7. Unsupported patterns of two disabled routers. (a) diagonalA(DA), (b) diagonalB(DB), (c) columnar(C), (d) row-at-edge(R).

The Pseudo code of the E-Rescuer routing algorithm is shown in Algorithm 1. Table 2 illustrates the required hop counts between source and destination for N, S, E, W, NE, NW, SE, SW-bound packets. If the packets are outside the distinct region, which means no disabled router is on paths, the required hop count maintains the same. If only intermediate disabled routers exist, the required hop count decreases.

4.4. Scaling and extension

Our solution is designed for the mesh topology. Each router has 7 ports, 2 ports per Y direction (North and South) and 1 port per X direction (East and West) and 1 port for the network interface. However, the design does not limit the cores' architecture. It can be used in SoCs with any types of cores and core sizes as long as such heterogeneity does not affect the mesh topology. Although the distinct region is 3 × 3, the Algorithm 1 also works for any network larger than or equal to 2 × 2 without any change.

4.5. Reliability analysis of multiple disabled routers

4.5.1. One disabled router

We have already proved that all single disabled routers are tolerated in the network without any packet loss and by allowing all cores to operate normally. Therefore by applying the proposed mechanism, all packets can be delivered to destinations in a presence of a disabled router.

4.5.2. Two disabled routers

Among all combinations of two disabled routers, only four patterns cannot be fully tolerated, shown in Fig. 7. For patterns with two neighboring routers in the same row, if one of them is located either at the eastern or western edge, it cannot be handled (row-at-edge in Fig. 7(d)). Otherwise, the patterns are supported. Patterns (a) and (b) may cause deadlock under high packet injection rates, as shown in Fig. 8. The south disabled router in Pattern (c),

Algorithm 1 The routing algorithm of E-Rescuer covering both inside and outside of distinct regions, *The value of X increases from west to east. The value of Y increases from north to south.

Require: Current: (X_C, Y_C); Source: (X_S, Y_S); Destination: (X_D, Y_D); Δ_X: $|X_D - X_C|$; Δ_Y: $|Y_D - Y_C|$; *in*: input port
Ensure: Selected port: *Sel*;

if $X_C = X_D$ and $Y_C = Y_D$ **then** *Sel* ←Local;
else if $X_C < X_D$ and $Y_C = Y_D$ **then**
  if Neighbor(E) is available **then** *Sel* ←E;
  else if Neighbor(E) is destination **then**
    if Neighbor(N) is available **then** *Sel* ←N1;
    else *Sel* ←S1;
    end if;
  else *Sel* ←E;
  end if;
else if $X_C > X_D$ and $Y_C = Y_D$ **then**
  if Neighbor(W) is available **then** *Sel* ←W;
  else if Neighbor(W) is destination **then**
    if Neighbor(N) is available **then** *Sel* ←N2;
    else*Sel* ←S2;
    end if;
  else *Sel* ←W;
  end if;
else if $X_C = X_D$ and $Y_C > Y_D$ **then**
  if Neighbor(N) is disabled **then** *Sel* ←N1;
  else if *in* is *S*1 **then** *Sel* ←N1;
  else if $X_D > X_S$ **then** *Sel* ←N1;
  else *Sel* ←N2;
  end if;
else if $X_C = X_D$ and $Y_C < Y_D$ **then**
  if Neighbor(S) is disabled **then**
    if Neighbor(S) is destination **then** *Sel* ←S2;
    else *Sel* ←S1;
    end if;
  else if *in* is *N*1 **then** *Sel* ←S1;
  else if $X_S > X_D$ **then** *Sel* ←S2;
  else *Sel* ←S1;
  end if;
else if $X_C < X_D$ and $Y_C > Y_D$ **then**
  if $\Delta_X = 1$ and $\Delta_Y = 1$ and destination router is disabled **then** *Sel* ←E;
  else *Sel* ←N1 or E according to congestion and availability;
  end if;
else if $X_C > X_D$ and $Y_C > Y_D$ **then**
  if $\Delta_X = 1$ and $\Delta_Y = 1$ and destination router is disabled **then** *Sel* ←W;
  else *Sel* ←N2 or W according to congestion and availability;
  end if;
else if $X_C < X_D$ and $Y_C < Y_D$ **then**
  if $\Delta_X = 1$ and $\Delta_Y = 1$ and destination router is disabled **then** *Sel* ←E;
  else *Sel* ←S1 or E according to congestion and availability;
  end if;
else if $X_C > X_D$ and $Y_C < Y_D$ **then**
  if $\Delta_X = 1$ and $\Delta_Y = 1$ and destination router is disabled **then** *Sel* ←W;
  else *Sel* ←S2 or W according to congestion and availability;
  end if;
end if;

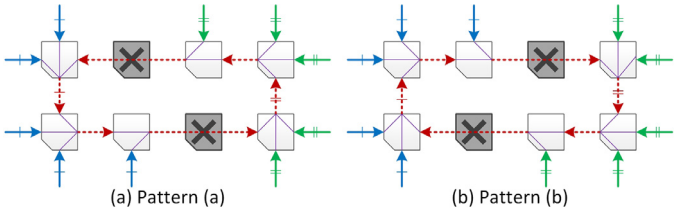

Fig. 8. Overlap of the paths leading to deadlock. The dash arrows show the deadlock ring. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

and the router at the edge in Pattern (d) cannot be accessed by other routers due to the configuration of bypassing channels.

The total number of two router combinations in an $n \times n$ mesh network can be measured by $N_{all-two} = \binom{n^2}{2}$. The number of diagonalA (DA), diagonalB (DB), columnar (C) and row-at-edge (R) combinations in an n× n mesh network can be calculated by:

$$N_{DA} = N_{DB} = (n-1)^2 \tag{1a}$$

$$N_C = n(n-1) \tag{1b}$$

$$N_R = 2n \tag{1c}$$

Thus, the ratio of the supported fault patterns over the total number of fault patterns with two disabled routers can be calculated by:

$$\begin{aligned} R_{2F} &= 1 - \frac{N_{DA} + N_{DB} + N_C + N_R}{N_{all-two}} \\ &= 1 - \frac{6n^2 - 6n + 4}{n^4 - n^2} \end{aligned} \tag{2}$$

According to this formula, the obtained value in 8 × 8 mesh networks is 91.57%. In other words, with this probability, the network works normally with two disabled routers without disabling any core and dropping any packet.

It is worth mentioning that among the unsupported patterns, shown in Fig. 7, in patterns (c) and (d) by disabling the cores connected to the disabled router, the network can still access all the other cores and remain deadlock free. In this paper, we focused only on the cases when all cores are kept alive in the system and thus disabling a core has not been investigated.

4.5.3. Three disabled routers

The unsupported patterns with three disabled routers consist of the unsupported patterns with two disabled routers (DA, DB, C, R) and any single disabled router (S) in the network (called DAS, DBS, CS, and RS). The number of unsupported two-disabled-router patterns is already calculated while the third disabled router cannot occur again in those two disabled routers. Thereby, the total number of unsupported three-disabled-router patterns can be calculated by multiplying the number of unsupported two-disabled-router patterns and the number of locations for the third disabled router as:

$$N_{DAS} = N_{DBS} = N_{DA} \times N_S = (n-1)^2\left(n^2 - 2\right) \tag{3a}$$

$$N_{CS} = N_C \times N_S = n(n-1)\left(n^2 - 2\right) \tag{3b}$$

$$N_{RS} = N_R \times N_S = 2n\left(n^2 - 2\right) \tag{3c}$$

As shown in Fig. 9, some unsupported patterns are calculated twice, and they should be removed. For example, the combination of a diagonalA and a single disabled router may result in the

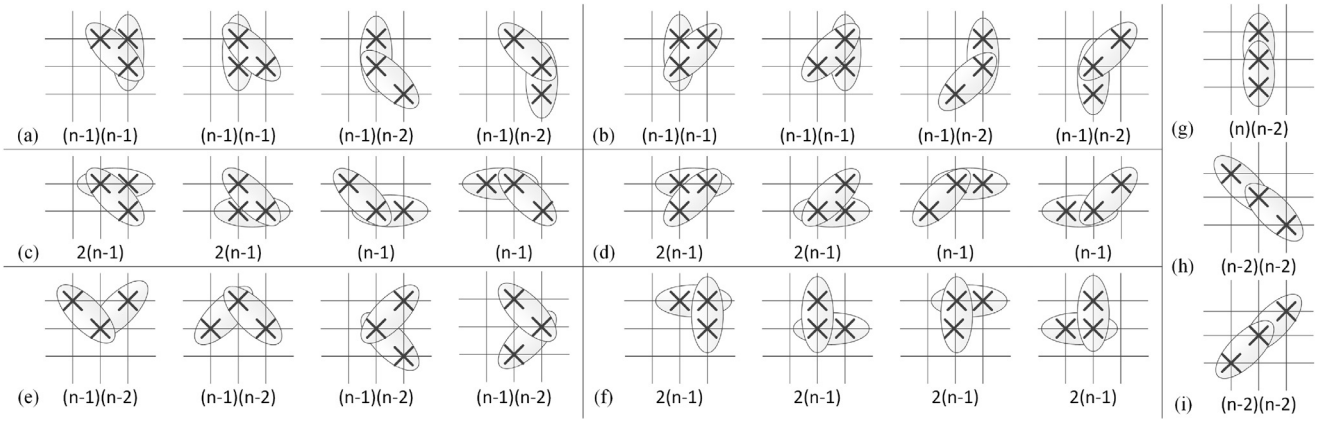

Fig. 9. Correlation (a) DA.C (b) DB.C (c) DA.R (d) DB.R (e) DA.DB (f) R.C (g) R.R (h) DA.DA (i) DB.DB (The formula under each figure shows the number of such patterns in a n× n network).

same pattern as the combination of a columnar and a single disabled router (Fig. 9(a)). Moreover, patterns in Fig. 9(f) are calculated triple times in Fig. 9(a–d). Therefore, they are not included in repeated patterns. For each repeated pattern, we measured the number of locations where that pattern may happen in an $n \times n$ network. The formulas are given bellow each pattern in Fig. 9. Thereby, the total number of repeated patterns can be measured by Formula (4).

$$\begin{aligned} N_{RP} &= N_{DA.C} + N_{DB.C} + N_{DA.R} + N_{DB.R} \\ &\quad + N_{DA.DB} + N_{R.R} + N_{DA.DA} + N_{DB.DB} \\ &= 15n^2 - 30n + 16 \end{aligned} \tag{4}$$

$$\begin{aligned} R_{3F} &= 1 - \frac{N_{DAS} + N_{DBS} + N_{CS} + N_{RS} - N_{RP}}{N_{all-three}} \\ &= 1 - \frac{6\left(3n^4 - 3n^3 - 19n^2 + 36n - 20\right)}{n^6 - 3n^4 + 2n^2} \end{aligned} \tag{5}$$

According to Formula (5), in an 8 × 8 mesh network, the ratio of the supported fault patterns over the total number of fault patterns with three disabled routers for the E-Rescuer algorithm is 76.47%.

For patterns with more than three disabled routers, if the pattern does not contain two routers, similar to those shown in Fig. 7, the pattern is supported. Otherwise, deadlock may occur, or the disabled routers are not accessible. The equations for more than three disabled routers are more complex and out of the scope of this work.

5. Results and discussion

In this section, the proposed E-Rescuer algorithm and other four comparative routing algorithms are implemented in a simulator. Based on the simulation results, the fault-tolerant capacity of E-Rescuer for different number of disabled routers is discussed first. Then, the performance under different traffic profiles, different numbers of disabled routers, different network sizes are provided and discussed. Moreover, the traffic distribution of three patterns is presented to demonstrate the benefit of the district region. Finally, the area and power of one router with different routing algorithms are reported.

5.1. Experimental setup

To evaluate the efficiency and performance of E-Rescuer, a mesh network is simulated using one visualization Network-on-Chip design framework, called *VisualNoC* [26,27]. Each router in this network has seven physical ports (one port for each of the local, east and west directions and two ports for each of the north and south directions) with 12-flit buffers at each physical port. During each simulation run, the first 2000 packets warm up the network and are not used in the statistics. After that, 30,000 packets are counted for performance statistics.

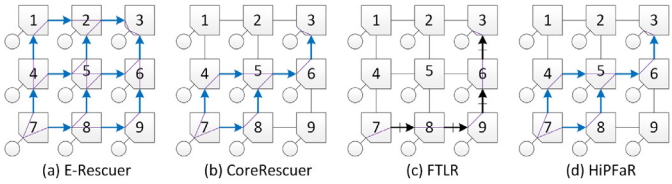

Fig. 10. Possible paths from Router 7 to Router 3. No router is disabled.

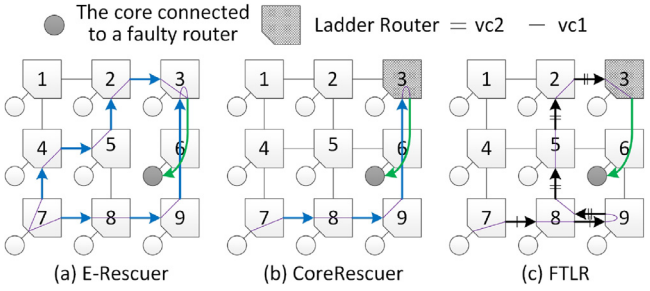

Fig. 11. Possible paths from Router 7 to Router 6. Router 6 is disabled.

We implemented the E-Rescuer, CoreRescuer [9], DyXY [28], FTLR [14] and HiPFaR [20] algorithms in our architecture. DyXY is a well accepted adaptive routing algorithm with traffic balance but not fault-tolerant. It is only used to compare the performance in a fault-free network. Fig. 10 shows the difference between different routing algorithms in a fault-free network while Fig. 11 shows an example where the destination router 6 is disabled; packets reach the destination node by routing through the ladder router.

HiPFaR is a high-performance fault-tolerant algorithm that uses shortest paths. HiPFaR disconnects the core from the network along with the disabled router. If there is no disabled router in the network, packets are adaptively routed to the southwest neighbor of the destination node and then to the southern neighbor and finally to the destination node, as shown in Fig. 10(d). However, since HiPFaR cannot utilize the bypassing channels, the packets from or to the disabled router are dropped. Therefore, during simulations, there is no packet with the disabled routers as source

Table 3
Traffic profiles used in simulation.

Uniform	Each destination router is chosen randomly.
Transpose1	(x,y) → (7−y,7−x).
Transpose2	(x,y) → (y,x).
BitReversal	Destination id is the bit reversal of source id.
Suffle	Destination id is the right loop shift of source id.
Butterfly	Get destination id by swapping the LSB and MSB of source id.

or destination. DyXY and HiPFaR require the same number of channels as E-Rescuer. The packets are assigned to the subnetwork A and the subnetwork B similar to E-Rescuer.

FTLR is a fault-tolerant routing algorithm aimed at tolerating complex faulty-link patterns. Packets travel around faulty links to reach destinations. To adapt it toward the proposed architecture, a faulty router is considered as a router with three faulty links while the links to the ladder router and network interface are assumed to be available. This assumption is made to employ the bypassing channels in FTLR. Thereby, if the destination of the packet is the core connected to the faulty router, the packet can reach the core using the bypassing channels. In other cases, the packet has to travel around the faulty router by passing through the healthy routers. As shown in Fig. 10(c), packets are routed based on the XY routing algorithm if there is no disabled router on the path. After facing a disabled router, a U-turn is taken to route a packet through another fault-free path (Fig. 11(c)). FTLR needs one more virtual channel on east and west directions to deliver the packets traveling around the faulty router.

CoreRescuer is the predecessor of E-Rescuer, and its major shortage is the limited adaptivity of packet routes. E-Rescuer increases adaptivity by limiting the effect of the disabled routers to a small region. In Fig. 10(a) and (b), if there is no disabled router in the network, E-Rescuer can choose all the shortest path between source and destination according to the congestion information. However, CoreRescuer routes the packets to the southwestern neighbor of the destination node adaptively and then to the southern neighbor and finally to the destination node. In the network with one disabled router (Fig. 11(a) and (b)), E-Rescuer still provides multiple paths from the source to the destination as illustrated in the figures.

For performance analysis, six different traffic profiles (Table 3) are used. These standard traffic profiles are used in many publications to compare different solutions.

5.2. Reliability analysis

The uniform traffic profile is used for reliability evaluation as it covers the packet transmission from any core to any other core in the network. To evaluate the reliability of E-Rescuer, the number of disabled routers increases from one to three. For extracting the reliability based on the number of supported patterns, all patterns of disabled routers are examined in the simulation, meaning that all possible combinations of two disabled routers (i.e. 2016) and three disabled routers (i.e. 41,664) in an 8 × 8 mesh network are verified. FTLR is also simulated to compare with E-Rescuer as it can also access the cores of the disabled routers. HiPFaR has not been taken into account in this section because it cannot handle packets from/to the cores connected to a disabled router. Simulations are run with the packet injection rate of 0.1 packets/cycle/router to provide sufficient traffic load for deadlocks to appear.

First, we evaluate the ratio of the supported fault patterns over the total number of fault patterns. A pattern is counted as a supported pattern if all packets reach their destinations. In other words, even a single packet drop renders the pattern unsupported. As illustrated in Fig. 12, all three algorithms are 100% reliable when there is a single disabled router in the network. When disabling

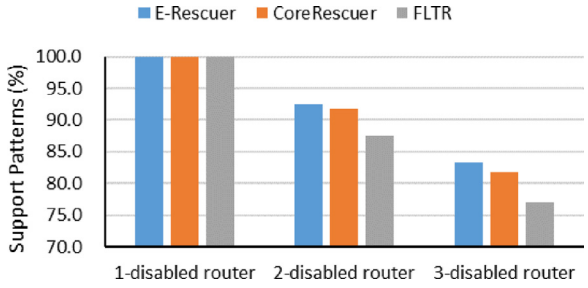

Fig. 12. The percentage of supported patterns for different number of disabled routers.

two routers, E-Rescuer and CoreRescuer can tolerate 92.56% and 91.82% of all combinations, respectively. While FTLR tolerates two disabled routers with the probability of only 87.55%, E-Rescuer tolerates about 5% more patterns (i.e. 101 patterns) than FTLR.

The simulation results reveal a better value than what has been obtained with analytic formulas (i.e. 91.57% patterns by analysis versus 92.56% patterns by simulation results). The reason for this difference is that some deadlock situations are so rare that they do not show up during the simulation. The occurrence of a deadlock does not only depend on the positions of disabled routers but also on the local congestion, the size of packets, and the state of buffers.[1] One example is recorded in the video, showing the case when the router 3 (row 0 column 3) and the router 7 (row 1 column 2) are disabled in a 5 × 5 network (coordinates start at 0, top-left) but no deadlock is observed under the uniform random traffic. Because the buffers on the deadlock cycle do not fill up at the same time. After we change the traffic distribution and increase the traffic in the deadlock cycle, the deadlock finally occurs. It is worth mentioning that uniform traffic is commonly used to extract reliability values in fault-tolerant designs. However, as our analysis shows, simulations with this traffic pattern do not necessarily identify all possible cases of cycles. Other synthetic traffic patterns underestimate the reliability problems even more.

Theoretically, CoreRescuer should have the same fault-tolerant capacity as E-Rescuer. Due to the reason explained above, more unsupported patterns are detected during the simulation of CoreRescuer. 18 patterns with disabled routers as Fig. 7(b) do not show deadlock and work correctly under E-Rescuer algorithm. Under CoreRescuer algorithm, only 5 patterns as Fig. 7(b) survive from deadlock. Therefore, E-Rescuer shows a higher ratio of supported patterns than CoreRescuer.

When three routers are knocked out, the reliability of E-Rescuer, CoreRescuer, and FTLR decreases to 83.25%, 81.78%, and 77.06% respectively.

We also measured the average number of successful packets' arrival at destinations as a fraction of the total number of injected packets. The average is taken over all combinations of patterns regardless of whether the pattern is supported or not. As can be seen from the results shown in Fig. 13, on average E-Rescuer successfully delivers 100%, 99.88%, and 99.63% of packets to their destinations under one, two and three disabled routers, respectively. For one, two and three disabled routers, the ratio of the successful packet arrival over the total number of delivered packets for the CoreRescuer algorithm is 100%, 99.83%, and 99.51%, while the ratio is 100%, 99.68%, and 99.04% for the FTLR algorithm.

[1] For a visualization of this observation, a video is available online: https://www.youtube.com/watch?v=5oDRC6WZEpI. It provides two cases where deadlock occurs but do not appear under a uniform random traffic simulation. The first example shows a general case and is not exactly the same as E-Rescuer while the second example is supported by E-Rescuer.

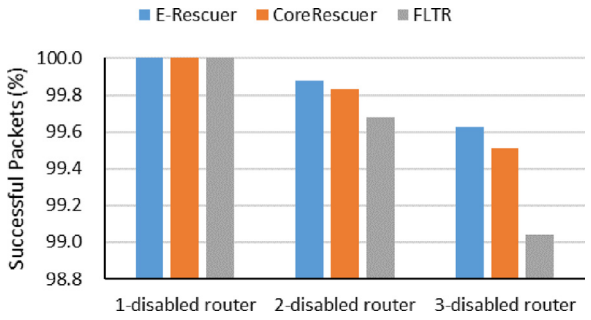

Fig. 13. The percentage of successful packets for different number of disabled routers.

5.3. Performance analysis of different traffic profiles

This section discusses the performance of 8 × 8 networks with 1 disabled router under different traffic profiles. The simulation result is averaged over 64 patterns. For each network, traffic is injected according to 6 different traffic patterns as listed in Table 3 and ten different packet injection rates. Figs. 14 and 15 illustrate the average latency and throughput, respectively.

Fig. 14 shows that E-Rescuer achieves lower latency than CoreRescuer, HiPFaR, and FTLR under most traffic patterns. There are several reasons for this improvement. FTLR uses DoR-XY as the default routing, and it cannot take advantage of bypassing channels. In HiPFaR and CoreRescuer, the adaptivity of all packets is affected by the single disabled router, leading to traffic congestion. E-Rescuer provides more adaptivity and chooses paths by considering congestion values. This adaptivity is available as long as the packets are outside of the distinct region. In other words, a disabled router affects the traffic only in the distinct region. Using bypassing channels helps to reduce the latency as they work similar to the shortcut channels. Therefore, E-Rescuer achieves better traffic balance than the other three methods in the presence of disabled routers.

The throughput results are shown in Fig. 15, following the same trend as the latency. Quantitatively, E-Rescuer enhances throughput up to 79.83%, 45.93%, 33.36% as compared to FTLR, HiPFaR, and CoreRescuer, respectively, for the example case when the injection rate is 0.07 packets/cycle/router and the traffic is bit-reversal. The performance improvements have a relationship with the traffic distribution. Under uniform traffic profile, the saturation throughput of E-Rescuer is 1.22 ×, 1.16 ×, and 1.28 × higher than the saturation throughput of HiPFaR, FTLR, and CoreRescuer, respectively. In the simulations under the Shuffle traffic profile, E-Rescuer has higher throughput than CoreRescuer and FTLR, but lower throughput than HiPFaR.

5.4. Performance analysis for different number of disabled routers

In this section, average latency and throughput are used to measure the performance of E-Rescuer, CoreRescuer, DyXY, HiPFaR and FTLR. For each algorithm, the performance is measured in the networks with 0 disabled routers, 1 disabled router (average among 64 patterns) and 2 disabled routers (average among 2016 patterns), but DyXY is only simulated in a fault-free network as it is not fault-tolerant. For each network, traffic is injected according to 6 different traffic patterns (as listed in Table 3) and ten different packet injection rates. Due to a very long simulation time, the networks with more than 2 disabled routers are not evaluated. Figs. 16 and 17 illustrate the average latency and throughput, respectively.

Figs. 16(a) and 17(a) show similar curves for E-Rescuer and DyXY. The reason is that both methods act similarly by offering a full adaptivity and choosing less congested paths in the disabled-router-free network.

Figs. 16 and 17 show that E-Rescuer achieves lower latency and a higher throughput than CoreRescuer, HiPFaR, and FTLR. Moreover, disabled routers increase the packet latency, and the network saturates earlier. Also as more routers are disconnected, the network saturates faster. By disconnecting more routers, the throughput ratio of E-Rescuer increases. For example, from 0 to 2 disabled routers, the ratios of throughput improvement are 23.11%, 27.63%, and 30.09% compared to the throughput of CoreRescuer. The reason is that distinct regions limit the impact of disabled routers.

5.5. Performance analysis of different network sizes

Fig. 18 illustrates the average saturation throughput in 4 × 4, 6 × 6, 8 × 8 and 10 × 10 networks. Traffic is injected according to uniform traffic, and the traffic load is enough to saturate the network. The actual value of packet injection rate is marked under X axis in Fig. 18. There is one disabled router in the network, and the throughput is averaged over all patterns. In the simulation, the network is already saturated and the value of latency is very large and not meaningful, so that only throughput is shown in Fig. 18. In all networks, E-Rescuer shows a significant improvement in throughput.

With increasing the network size, E-Rescuer exhibits a stable throughput improvement of over 20%. As Fig. 18 shows, the throughput benefit of E-Rescuer is between 24% and 30%. As the proposed method does not have a limitation on the network size, it is expectable that the throughput will continue increasing as the network size enlarges. Also E-Rescuer achieves a better throughput than the other methods.

5.6. Traffic balance

Faults lead to hotspot regions in the network. Adaptivity can play an important role to avoid performance drops. As we have discussed in Section 5. A, E-Rescuer provides more paths between a given source and destination as compared to CoreRescuer and HiPFaR. Thereby, packets have more path choices to select from based on the congestion information that consequently lead to a better traffic balance. FTLR has the lowest adaptivity because it is based on XY routing.

To illustrate the benefits introduced by the distinct region, Fig. 19 shows an example of the load distribution for three patterns of 1, 2, and 3 disabled routers in a 9 × 9 NoC. The traffic injection rate is 0.05 packets/cycle/router under uniform random traffic, and each router injects 500 packets of length 5 flits during the simulation. Both algorithms, CoreRescuer and E-Rescuer, lead to hot spots around the disabled router. The northwest, northeast, southwest, and southeast neighbor routers have higher traffic than other routers.

However, a close examination shows that the maximum traffic of E-Rescuer is lower than the maximum traffic of CoreRescuer, and the minimum traffic of E-Rescuer is higher than that of CoreRescuer. Thus, the traffic range of E-Rescuer is smaller. The traffic of E-Rescuer is more balanced than that of CoreRescuer due to two reasons. First, the disabled router handles more traffic in the E-Rescuer configuration. Second, the E-Rescuer algorithm is more adaptive than CoreRescuer even if no router is disabled, which leads to a better load distribution and explains that the peripheral regions handle more traffic in the right-hand plots of Fig. 19 than in the left-hand plots. Together these two effects lead to a better traffic distribution.

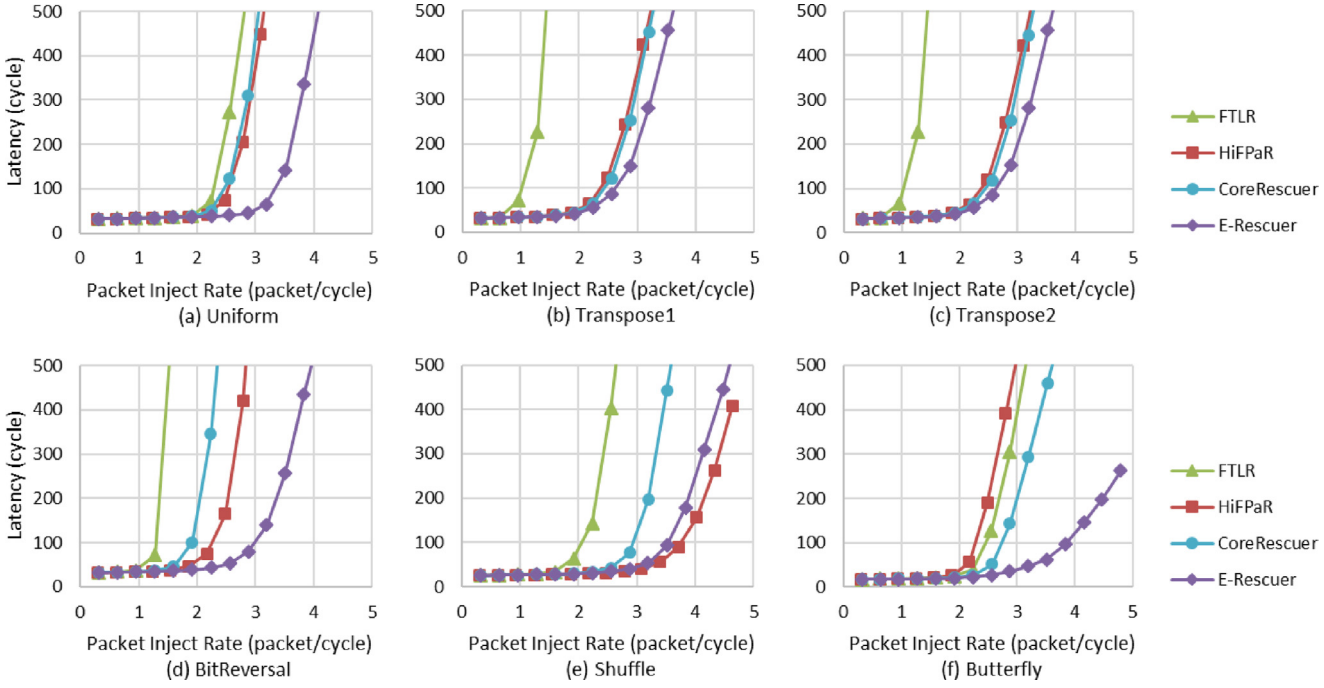

Fig. 14. Average latency under different traffic pattern in 8 × 8 mesh network with one disabled router.

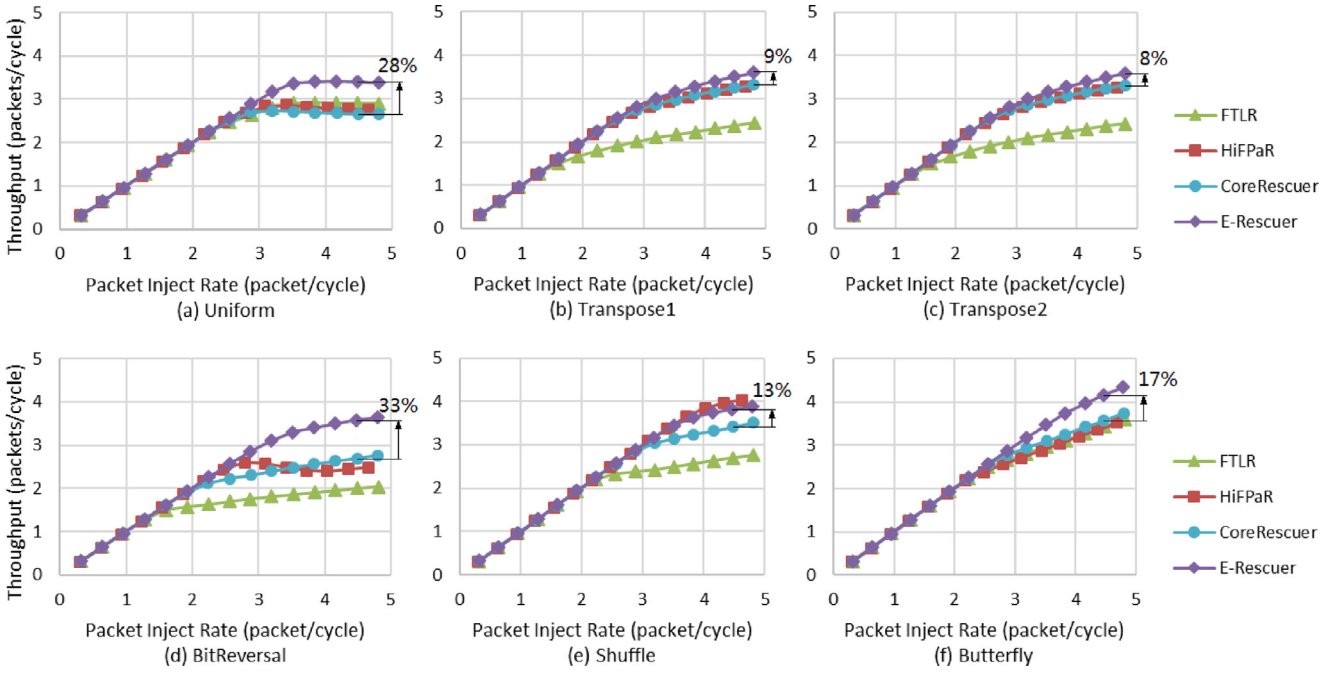

Fig. 15. Throughput under different traffic pattern in 8 × 8 mesh network with one disabled router.

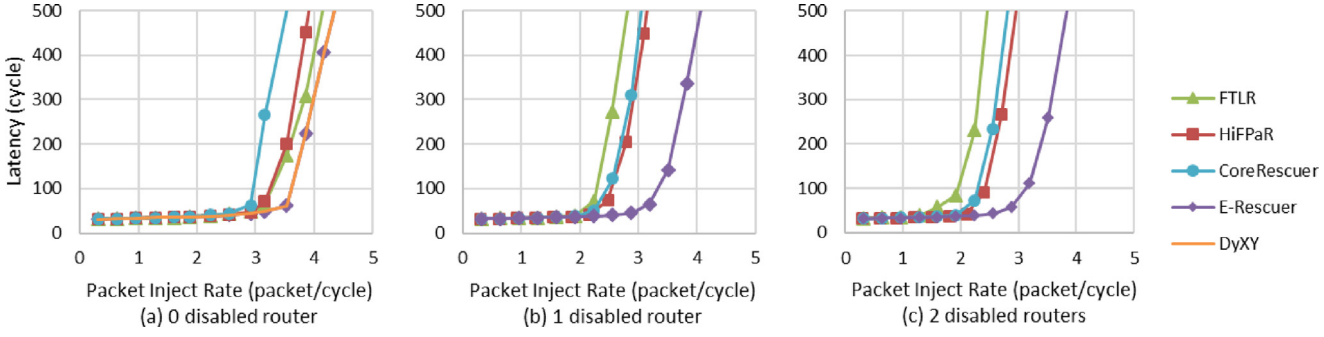

Fig. 16. Average latency under uniform traffic pattern in 8 × 8 mesh network with different number of disabled routers.

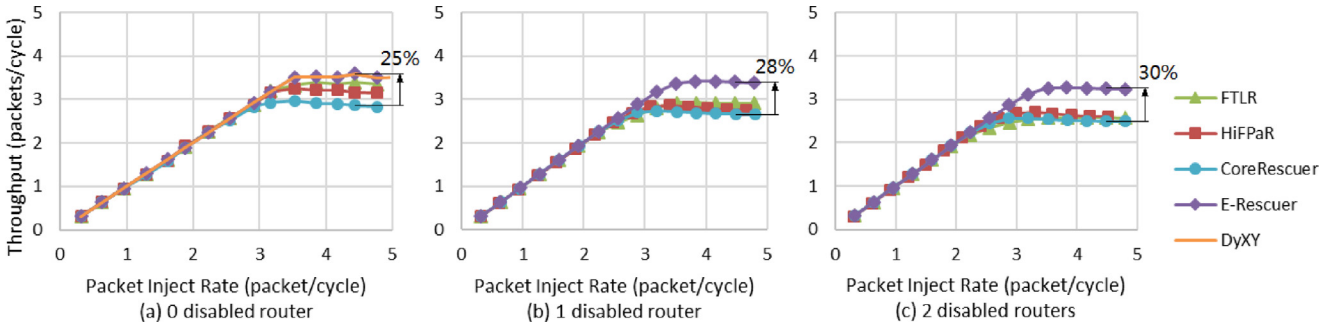

Fig. 17. Throughput under uniform traffic pattern in 8 × 8 mesh network with different number of disabled routers.

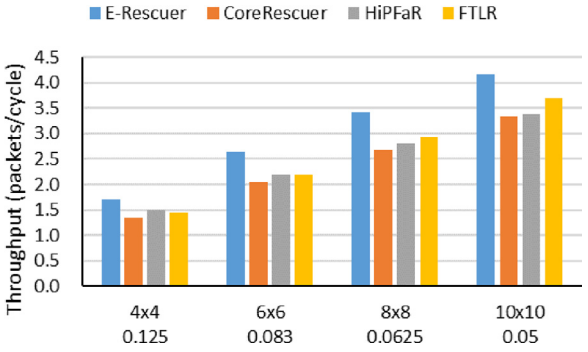

Fig. 18. Saturation throughput for different network sizes. The packet injection rate of each network is marked below the network size. The unit is packet/cycle/router.

5.7. Hardware analysis

To assess the area overhead and power consumption of different router architectures and routing algorithms, four different configurations are synthesized using Synopsys Design Compiler. The baseline router has no bypassing channels and runs the DyXY routing algorithm. It also has 7 ports and 1 VC per port. E-Rescuer, CoreRescuer, and FTLR routers employ bypassing channels and their routing algorithms. We used TSMC45nm technology at the operating frequency of 1 GHz and supply voltage of 0.9 V. As shown in Table 4, bypassing channels infrastructure increases area by about 7% compared with the baseline router architecture. From this 7%, the contribution of bypassing channels is 2.54% while the rest comes from the implementation of the routing algorithm. The power consumption and area overhead of E-Rescuer and CoreRescuer are lower than those of FTLR. The reason is that FTLR employs extra virtual channels on the east and west directions and a more complex routing algorithm. Moreover, compared with CoreRescuer, E-Rescuer still reduces area and power because of the simplified routing algorithm.

Table 4
Power and area analysis.

	Baseline	CoreRescuer	E-Rescuer	FTLR
(a) One Router				
Area (μm²)	23,925.3	25,905.2	25,617.5	33,633.1
Power (mW)	1.351	1.413	1.403	1.968
(b) Bypass channels including Mux and Demux				
Area (μm²)	–	651.3	651.3	–
Power (mW)	–	0.023	0.023	–
(c) One Routing Calculator				
Area (μm²)	109.0	298.8	257.7	563.6
Power (mW)	0.010	0.016	0.014	0.035

* Fault detection units are not included. All methods share the same fault detection method.

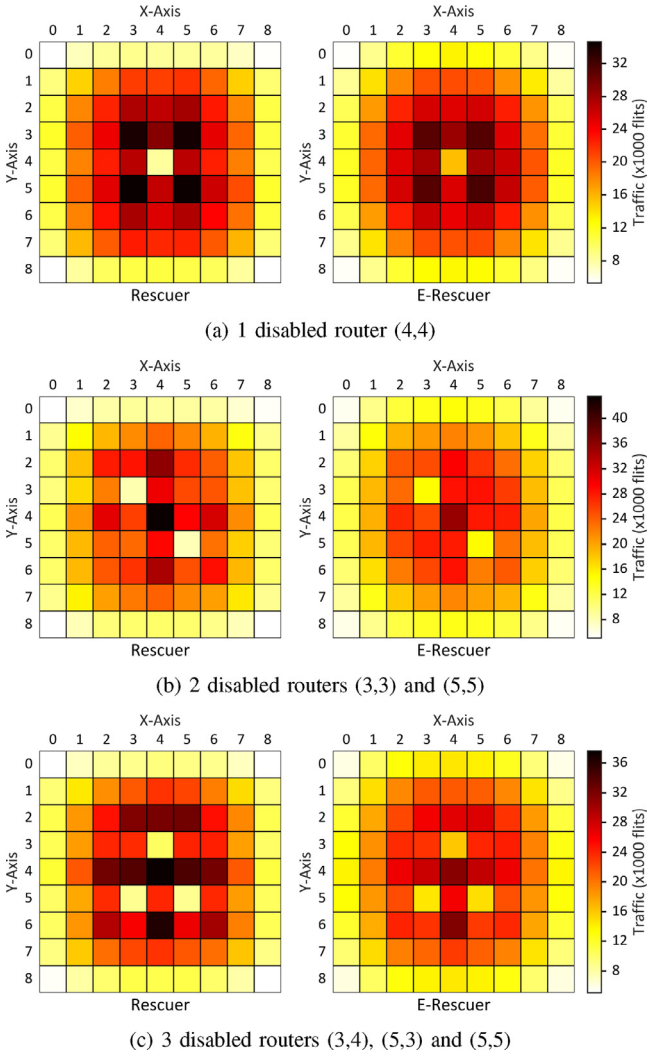

Fig. 19. Color coded traffic distribution of CoreRescuer and E-rescuer in a 9 × 9 NoC; white denotes low load and black means high load.

6. Conclusion

Due to faults in routers or for testing purposes it is desirable to be able to tolerate disabled routers. In traditional methods, a disabled router leads to cutting off the core from the rest of the network. However, this is a waste and may have a severe impact on the functionality or performance of the entire system. In this paper, the E-Rescuer mechanism is proposed to allow the cores to send/receive packets to/from all cores in the network without any

packet loss in the presence of disabled routers. E-Rescuer suggests a reconfigurable router architecture and an adaptive fault-tolerant routing algorithm. By simply bypassing the disabled router in horizontal and vertical directions, E-Rescuer avoids taking non-minimal routes to bypass disabled routers and thus reducing the latency. Moreover, the adaptive routing algorithm provides more freedom in choosing paths outside the 3-by-3 area centered by a disabled router, which results in a more balanced traffic and better performance.

Simulation results demonstrate that E-Rescuer can achieve a better reliability and performance as compared to CoreRescuer, HiPFaR, and FTLR. E-Rescuer tolerates 100%, 92.56% and 83.25% of patterns with 1, 2, and 3 disabled routers. Moreover, E-Rescuer provides up to 80%, 46% and 33% better throughput than FTLR, HiPFaR, and CoreRescuer, respectively.

In the future, we will work on this reconfigurable router architecture further and propose fault-tolerant routing algorithms which can tolerate disabled links.

Acknowledgment

The authors would like to thank the anonymous reviewers for their constructive and helpful suggestions and comments. This paper was supported by the National Natural Science Foundation of China under grant (NSFC) No.61534002, No.61471407, the Chinese National Program for Support of Top-Notch Young Professionals (1st Batch), the Oversea Academic Training Funds (OATF), UESTC and Chinese Scholarship Council (CSC). This work is also supported by Academy of Finland and VINNOVA-MarieCurie.

References

[1] M. Kakoee, V. Bertacco, L. Benini, At-speed distributed functional testing to detect logic and delay faults in NoCs, IEEE Trans. Comput. 63 (3) (2014) 703–717.
[2] M. Radetzki, C. Feng, X. Zhao, A. Jantsch, Methods for fault tolerance in networks-on-chip, ACM Comput. Surv. 46 (1) (2013) 8:1–8:38.
[3] J. Wang, M. Ebrahimi, L. Huang, A. Jantsch, G. Li, Design of fault-tolerant and reliable networks-on-chip, in: IEEE Computer Society Annual Symposium on VLSI (ISVLSI), IEEE, 2015, pp. 545–550.
[4] M. Palesi, M. Daneshtalab, Routing algorithms in networks-on-chip, Springer, 2014.
[5] M. Ali, M. Welzl, S. Hessler, A fault tolerant mechanism for handling permanet and transient failures in a network-on-chip, in: Information Technology, 2007 4th International Conference on, IEEE, 2007, pp. 1027–1032.
[6] C. Feng, Z. Lu, A. Jantsch, J. Li, M. Zhang, FoN: fault-on-neighbor aware routing algorithm for networks-on-chip, in: SOC Conference, 2010 IEEE International, IEEE, 2010a, pp. 441–446.
[7] C. Feng, Z. L, A. Jantsch, J. Li, M. Zhang, A reconfigurable fault-tolerant deflection routing algorithm based on reinforcement learning for network-on-chip, in: the Third International Workshop on Network on Chip Architectures, ACM, 2010b, pp. 441–446.
[8] M. Pirretti, G. Link, R. Brooks, N. Vijaykrishnan, M. Kandemir, M. Irwin, Fault tolerant algorithms for network-on-chip interconnect, in: VLSI, 2004 IEEE Computer Society Annual Symposium on, IEEE, 2004, pp. 46–51.
[9] M. Ebrahimi, J. Wang, L. Huang, M. Daneshtalab, A. Jantsch, Rescuing healthy cores against disabled routers, in: IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), IEEE, 2014, pp. 98–103.
[10] D. Fick, A. DeOrio, G. Chen, V. Bertacco, D. Sylverster, D. Blaauw, A highly resilient routing algorithm for fault-tolerant NoCs, in: Design, Automation & Test in Europe Conference & Exhibition, 2009, IEEE, 2009, pp. 21–26.
[11] K. Aisopos, A. DeOrio, L. Peh, V. Bertacco, Ariadne: agnostic reconfiguration in a disconnected network environment, in: Parallel Architectures and Compilation Techniques, 2011 International Conference on, IEEE, 2011, pp. 298–309.
[12] H. Hsin, E. Chang, C. Lin, A.-Y. Wu, Ant colony optimization-based fault-aware routing in mesh-based network-on-chip systems, IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. 33 (11) (2014) 1693–1705.
[13] R. Bishnoi, V. Laxmi, M. Gaur, J. Flich, d^2-lbdr: distance-driven routing to handle permanent failures in 2d mesh NoCs, in: Design, Automation Test in Europe Conference Exhibition (DATE), IEEE, 2015, pp. 800–805.
[14] A. Vitkovskiy, V. Soteriou, C. Nicopoulos, Dynamic fault-tolerant routing algorithm for networks-on-chip based on localised detouring paths, IET Comput. Digital Tech. 7 (2) (2013) 63–103.
[15] C. Chen, S. Cotofana, Towards an effective utilization of partially defected interconnections in 2d mesh Nocs, in: IEEE Computer Society Annual Symposium on VLSI (ISVLSI), IEEE, 2014, pp. 492–497.
[16] M. Gmez, J. Duato, J. Flich, P. Lpez, A. Robles, N. Nordbotten, O. Lysne, T. Skeie, An efficient fault-tolerant routing methodology for meshes and tori, Comput. Archit. Lett. 3 (1) (2004) 3.
[17] Z. Zhen, A. Greiner, S. Taktak, A reconfigurable routing algorithm for a fault--tolerant 2d-mesh network-on-chip, in: Design Automation Conference, 2008, 45th ACM/IEEE, IEEE, 2008, pp. 441–446.
[18] A. DeOrio, D. Fick, V. Bertacco, D. Sylvester, D. Blaauw, J. Hu, G. Chen, A reliable routing architecture and algorithm for NoCs, IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. 31 (5) (2012) 726–739.
[19] L. Chen, T. Pinkston, Nord: node-router decoupling for effective power-gating of on-chip routers, in: Microarchitecture, 2012 45th Annual IEEE/ACM International Symposium on, IEEE, 2012, pp. 270–281.
[20] M. Ebrahimi, M. Daneshtalab, J. Plosila, High performance fault-tolerant routing algorithm for NoC-based many-core systems, in: Parallel, Distributed and Network-Based Processing (PDP), 2013 21st Euromicro International Conference on, IEEE, 2013, pp. 462–469.
[21] B. Fu, Y. Han, H. Li, X. Li, Zonedefense: a fault-tolerant routing for 2-d meshes without virtual channels, IEEE Trans. Very Large Scale Integr.(VLSI) Syst. 22 (1) (2014) 113–126.
[22] F. Chaix, D. Avresky, N. Zergainoh, M. Nicolaidis, Fault-tolerant deadlock-free adaptivve routing for any set of link and node failures in multi-cores systems, in: Network Computing and Application, 2010 9th IEEE International Symposium on, IEEE, 2010, pp. 52–59.
[23] J. Flich, S. Rodrigo, J. Duato, An efficient implementation of distributed routing algorithms for NoCs, in: 2nd IEEE/ACM International Symposium on Networks on Chip (NoCS), IEEE, 2008, pp. 87–96.
[24] V. Rantala, T. Lehtonen, P. Liljeberg, J. Plosila, Multi network interface architectures for fault tolerant network-on-chip, in: International Symposium on Signals, Circuits and Systems, IEEE, 2009, pp. 1–4.
[25] I. Srinivasan, S. Adve, P. Bose, J. Rivers, The case for lifetime reliability-aware microprocessors, in: 31st Annual International Symposium on Computer Architecture, 2004, p. 276.
[26] J. Wang, Y. Huang, M. Ebrahimi, L. Huang, Q. Li, A. Jantsch, G. Li, Visualnoc: a visualization and evaluation environment for simulation and mapping, in: Proceedings of the Third ACM International Workshop on Many-core Embedded Systems, ACM, 2016, pp. 18–25.
[27] Visualnoc: visualization network-on-chip simulation environment, (http://ic.uestc.edu.cn/SWpage/index.php/ESY-Series_Many-core_System-on-Chip_Simulation_Environment).
[28] M. Li, Q. Zeng, W. Jone, Dyxy - a proximity congestion-aware deadlock-free dynamic routing method for network on chip, in: Design Automation Conference, 2006 43th ACM/IEEE, ACM/IEEE, 2006, pp. 849–852.