Computing (2015) 97:631–648
DOI 10.1007/s00607-013-0362-9

# A Light-weight fault-tolerant routing algorithm tolerating faulty links and routers

**Masoumeh Ebrahimi · Masoud Daneshtalab**

Received: 5 May 2013 / Accepted: 3 October 2013 / Published online: 25 October 2013

**Abstract** Faults at either the link or router level may result in the failure of the system. Fault-tolerant routing algorithms attempt to tolerate faults by rerouting packets around the faulty region. This rerouting would be at the cost of significant performance loss. The proposed algorithm in this paper is able to tolerate both faulty routers and links with negligible impact on the performance. In fact, the proposed algorithm avoids taking unnecessary longer paths and the shortest paths are always taken as long as a path exists. On the other hand, fault-tolerant routing algorithms might be based on deterministic routing in which all packets use a single path between each pair of source and destination routers. Using deterministic routing, packets reach the destination in the same order they have been delivered from the source so that no reordering buffer is needed at the destination. For improving the performance, fault-tolerant algorithms might be based on adaptive routing in which packets are delivered through multiple paths to destinations. In this case, packets should be reordered at the destinations demanding reordering buffers. The proposed algorithm can be configured in both working modes, such that it can be based on deterministic or adaptive routing.

**Keywords** Fault-tolerant · Reliability · In-order delivery · Deterministic and adaptive

**Mathematics Subject Classification** 68M15 · 68M10

M. Ebrahimi (✉) · M. Daneshtalab
Department of Information Technology, University of Turku, Turku, Finland
e-mail: masebr@utu.fi

M. Daneshtalab
e-mail: masdan@utu.fi

## 1 Introduction

Faults may occur in different components of a network as cores, links or routers. When a core is faulty, the connected router and links can continue functioning. So, the effect of faults is not propagated to other parts of the network. When a router or a link is faulty, not only the connected core may not send or receive packets but also packets from the other cores cannot be transmitted through this faulty router or link [1]. This may result in blocking of packets and thus the failure of the whole system. Therefore, special care should be taken when a router or link becomes faulty. The most common solution to address this problem is to reconfigure the routing algorithm around a faulty region and to reroute packets around this region. This implies a non-minimal routing algorithm which increases the latency of packets significantly.

In wormhole routing, messages are divided into small flits traversing within the network in a pipelined fashion. This approach eliminates the need to allocate large buffers in intermediate routers along the path [2]. Moreover, in wormhole routing, message latency is less sensible to distance. However, it should be used with special care to avoid deadlock in the network. Deadlock is a situation when packets continuously wait for each other to release resources. Routing algorithms are required to be deadlock-free and break all cyclic dependencies among channels. Virtual channels are mainly used in the network for avoiding deadlock, improving performance and tolerating faults, but they impose area overhead and complexity [3].

Routing techniques provide some degrees of fault tolerance and can be categorized into deterministic and adaptive [4–6]. In deterministic routing algorithms, a fixed path is used between each pair of routers. The simplest deterministic algorithm is dimension-order routing which is known as XY or YX. Implementations of deterministic routing algorithms are simple. However, sending packets through a single path may result in the network congestion. In adaptive routing algorithms, a packet is not restricted to a single path and it can adaptively choose among the available paths. So, adaptive algorithms can decrease the probability of routing packets through congested or faulty regions. Conventional fault-tolerant routing algorithms reroute packets around faulty regions, either convex or concave, so that the selected paths are not always the shortest ones. Rerouting is an expensive solution and considerably increases packet's latency and router's complexity. On the other hand, most fault-tolerant routing algorithms are focused either on tolerating faulty links or faulty routers and a general method to tolerate both types of faults has rarely been discussed in literature. In addition, fault-tolerant algorithms are mostly based on deterministic routing, degrading the performance of the network.

From the in-order delivery point of view, when a master core issues several requests (or a request consisting of several packets) to a memory, by using a deterministic routing algorithm all packets follow the same route to reach the memory, and thus responses arrive to the master core in the same order as well. On the other hand, when exploiting an adaptive routing algorithm, the responses return to the master core in different orders and therefore a reordering mechanism is needed to handle the ordering of packets [7,8].

In this paper, we present a general fault-tolerant routing algorithm **T**olerating both **F**aulty **L**inks and **R**outers, called TFLR. This algorithm tolerates one faulty link or router using the shortest paths between each pair of source and destination routers, if a shortest path exists. It can also tolerate multiple faults if they have enough distances from each other. TFLR can be configured to act as a deterministic or adaptive routing algorithm. Deterministic routing assures the in-order delivery of packets while adaptive routing improves the performance of the network. TFLR is very simple and can be easily implemented.

The reminder of this paper is organized as follows. In Sect. 2, the related work is discussed. In Sect. 3, the proposed fault-tolerant routing algorithm is introduced. The results are reported in Sect. 4 while the summary and conclusion are given in the last section.

## 2 Related work

Fault-tolerant routing algorithms can be classified into two main groups: one can handle convex or concave regions [9–11] and the other group utilizes the contour strategy for addressing faults [12,13]. The methods in the first group are based on defining fault ring (f-ring) or fault chain (f-chain) around faulty regions where healthy routers are disabled in order to form a specific shape. For example, Extended X-Y is a well-known routing algorithm presented in [14]. This algorithm is designed based on XY routing and the odd-even turn model [15]. Similar to the odd-even turn model, this algorithm is deadlock-free without using any virtual channels by prohibiting certain turns in odd and even columns. In fault-free cases and depending on the position of the source and destination routers, this algorithm may perform similar to the XY routing algorithm (minimal routing) or take a longer path (non-minimal routing). In more details, if the source router is located in an even column, the packet is sent to the Y dimension; otherwise it has to take a hop to the west direction to reach an even column before making a turn toward the Y dimension. The packet follows YX routing until it reaches the destination router. When the packet faces a fault along its path, it has to be routed around the fault based on some specific rules. Two examples of Extended X-Y are shown in Fig. 1a. As can be seen in this figure, packets have to take a very long path to bypass faults while they could be simply routed through the shortest paths. It is worth

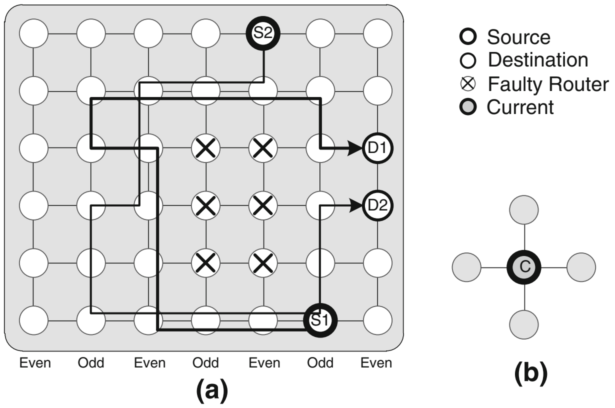

**Fig. 1** **(a)** Two examples of the Extended X-Y routing algorithm. **(b)** The required fault information

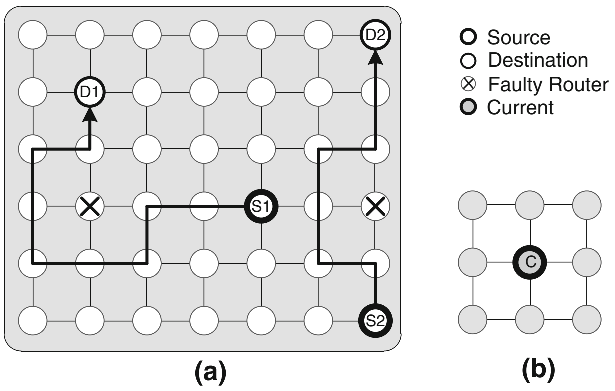

**Fig. 2** **(a)** Two examples of the ReRS routing algorithm. **(b)** The required fault information

mentioning that, this algorithm has many restrictions on the location of faults such as faults cannot be tolerated on borderline routers and there should be enough distance between faulty regions. Extended X-Y only needs to know about the fault statuses of its direct neighbors to make its routing decision (Fig. 1b). This algorithm is deterministic and does not make any effort toward alleviating congestion in the network.

A reconfigurable routing algorithm using the contour strategy provides the possibility of routing packets through a cycle-free contour surrounding a faulty component. For example, Z. Zhang et al. presented a reconfigurable routing algorithm [12] to tolerate a single faulty router in a mesh network without using virtual channels and disabling healthy routers. We call this **Re**configurable **R**outing **S**cheme, ReRS. To tolerate more number of faulty routers, the contours must not be overlapped and thus faulty routers should be located far away from each other. In other words, ReRS can tolerate a single faulty router in the network or multiple faulty routers if their contours do not overlap. This algorithm is deterministic and thus the performance cannot be improved by distributing packets over multiple paths. This method shows that cycles can naturally be avoided in borderline routers. Two examples of this method are shown in (Fig. 2a). Packets are routed normally inside the network using a deterministic method and they have to turn around a fault when facing to it. Each router should be informed about the fault statuses of eight direct and indirect neighboring routers (Fig. 2b).

ReRs has been extended in the RAFT method [13] to tolerate two faulty links. For this purpose, RAFT requires two virtual channels along both X and Y dimensions. This algorithm investigates a cycle-free contour for all combinations of one and two faulty links. This algorithm is very complicated and has many exceptional rules. Two examples of this algorithm are illustrated in Fig. 3a. RAFT is an adaptive method and is able to deliver packets through multiple paths. This reduces the latency as the probability of sending packets through the shortest paths increases. In RAFT, each router needs to know the fault statuses of twelve surrounding links (Fig. 3b).

A different fault-tolerant approach, called BFT-NoC, is presented in [16]. In this method, two channels are used for transmitting and receiving packets between two adjacent routers. In fault-free cases, one channel is dedicated for transmitting packets and another one for receiving packets. The idea of BFT-NoC is to share a channel for both transmitting and receiving packets when one of the channels is faulty. This

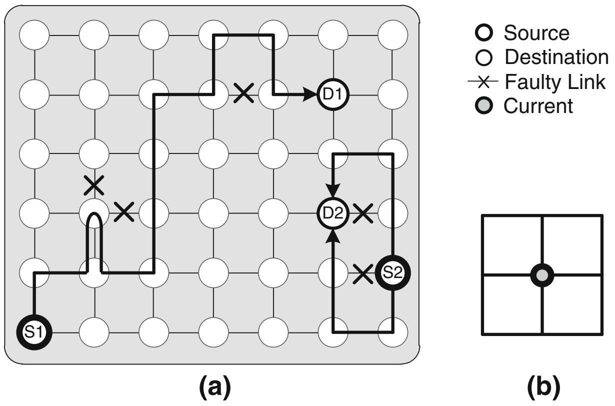

**Fig. 3** **(a)** Two examples of the RAFT routing algorithm. **(b)** The required fault information

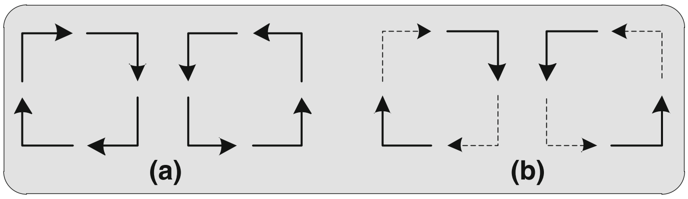

**Fig. 4** **(a)** Clockwise and counter-clockwise turns. **(b)** Permitted and prohibited turns in the XY routing algorithm (Note that dash lines indicate prohibited turns)

method reduces packets' latencies by avoiding costly rerouting of packets. By sharing the resources, the link performs as a bottleneck in high traffic loads. To reduce the load on the partially functioning links, it might be better to deliver packets through alternative routes. This is the solution that can be offered by routing algorithms. Obviously, BTF-NoC cannot be used to address faulty routers or a total link failure in both unidirectional directions.

Fault-tolerant routing algorithms could be also divided into two classes: the methods using virtual channels [13,17,18] and those without using virtual channels [14,19]. In general, different methods define a tradeoff between the number of virtual channels, the ability to handle different fault models, and the degree of adaptiveness. The virtual channel-based fault-tolerant routing algorithms provide better fault-tolerant characteristics than those without virtual channels. On the other hand, fault-tolerant routing algorithms are usually concentrated on two single point problems, tolerating faulty links [13,16,18,20] or tolerating faulty routers [12,14,21,22]. In this paper, we present a method and an algorithm which covers both types of faults.

## 3 The proposed fault-tolerant routing algorithm

### 3.1 Turn model

When different packets are routed inside the network, there is a possibility of forming two complete cycles, known as clockwise and counter-clockwise (Fig. 4a). A situation

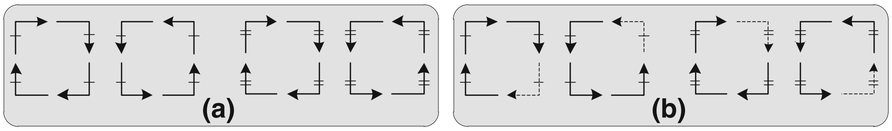

**Fig. 5** Permitted and prohibited turns of TFLR

is called deadlock when each packet in a cycle waits for another one to proceed and thus different packets block each other forever. In turn models, certain turns are prohibited from clockwise and counter-clockwise cycles in order to break all cyclic dependencies and thus avoiding deadlock. In the XY routing algorithm, for example, packets are first routed along the X dimension and then along the Y dimension. As shown in Fig. 4(b), in this algorithm, N-E (i.e. a packet moving to the north direction makes a turn to the east direction) and S-W (i.e. a packet moving to the south direction makes a turn to the west direction) turns cannot be taken from the clockwise cycle and N-W and S-E turns are not used from the counter-clockwise cycle. As a result, there is no possibility of forming a complete cycle among the remaining turns.

TFLR utilizes one and two virtual channels along the X and Y dimensions, respectively, in which four cycles might be formed in the network (Fig. 5a). In order to avoid deadlock, one turn is prohibited from each cycle which is shown in Fig. 5b. The prohibited turns in each virtual channel are taken from the Mad-y method [23]. Based on this turn model, the northeast packets can take E-N1, N1-E, and N2-E turns. For northwest packets, the turns W-N1, N2-W, and W-N2 are allowable. Southeast packets can take S1-E, E-S1, and S2-E turns. Finally, southwest packets can take W-S1, S1-W, and S2-W turns. In other words, for all positions of the source and destination routers, packets can take either X or Y dimension. This assures a fully adaptive routing algorithm. To prove deadlock-freeness, we use a numbering mechanism similar to the Mad-y method. This numbering mechanism shows that all the allowable turns have occurred only in an ascending order, and thus no cycle can be formed in the network. A two-digit number (a,b) is assigned to each input and output channel of a router in an $n \times m$ mesh network. According to the numbering mechanism, a turn connecting the input channel $(I_a, I_b)$ to the output channel $(O_a, O_b)$ is called an ascending turn when $(O_a > I_a)$ or $((O_a = I_a)$ and $(O_b > I_b))$. Figure 6 shows the channels numbering of a router at the position (X, Y). By using this numbering mechanism, it is guaranteed that all allowable turns are taken strictly in an increasing order, so that the TFLR routing algorithm is deadlock-free. For instance, if the turn E-N1 (i.e. a packet moving to the east direction makes a turn to the north direction using the first virtual channel) is taken into consideration, the west input channel with the label $(I_a = m+x, I_b = 0)$ is connected to the first virtual channel of the north output port having the label $(O_a = m + x, O_b = 1 + y)$. This turn takes place in an ascending order since $((O_a = I_a)$ and $(O_b > I_b))$. Similarly, all the other allowable turns by TFLR are taken in an ascending order. All allowable and unallowable turns of TFLR are listed in Table 1.

## 3.2 Tolerating faulty links and routers

A destination router might be located in eight different positions of a source router as north, south, east, west, northeast, northwest, southeast, and southwest. Since TFLR

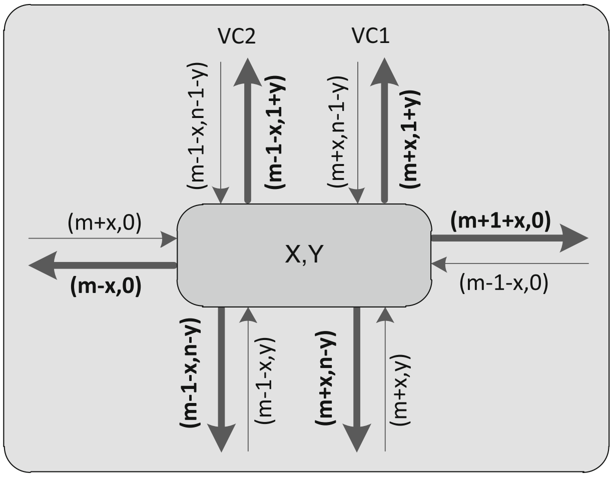

**Fig. 6** The numbering mechanism of TFLR

**Table 1** All allowable and non-allowable turns by TFLR

| Allowable turns | | Unallowable turns |
|---|---|---|
| E-N1 | N1-E | E-N2 |
| E-S1 | N2-E | E-S2 |
| W-N1 | N2-W | N1-W |
| W-N2 | S1-E | S1-W |
| W-S1 | S2-E | |
| W-S2 | S2-W | |

is based on a fully adaptive routing algorithm, all shortest paths are valid for packets. When there is a single fault in the network and the destination is in the northeast, northwest, southeast, and southwest positions of a source router, packets are able to use only the shortest paths to bypass the fault. Thereby, using TFLR, no rerouting takes place in these cases and the fault is bypassed prior reaching it. Obviously, for eastward, westward, northward, and southward packets, non-minimal paths must be taken if the shortest path is faulty. In sum, the fault is tolerated using non-minimal paths when the source and destination routers are located in the same row or column. In other cases, only the shortest paths are taken from the source to the destination router.

The example of Fig. 7 shows how the fault can be bypassed without rerouting packets when the destination is in the northeast position of the current router. As illustrated in Fig. 7a, the packet can be delivered to the north or east direction when the distances along both directions are one ($\Delta X = 1$ and $\Delta Y = 1$). If the northeast path is non-faulty (either the links or router), the packet will be sent through it. However, if one of the components in this path is faulty, the packet is routed through the east direction and could safely reach the destination (as we assume there is one fault in the network which is already bypassed). As a result, the packet is routed through the shortest path to the destination. In Fig. 7b, the distance along the X dimension has

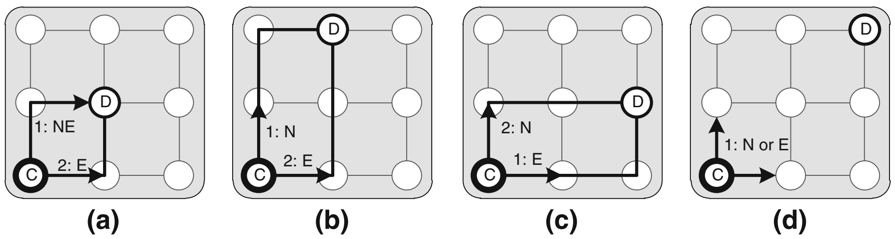

**Fig. 7** Bypassing faults when the destination is located in the northeast position of the source router (Note that numbers determine the priority of selecting among different routes)

reached one while the distance along the Y dimension is greater than one ($\Delta X = 1$ and $\Delta Y \geq 2$). According to TFLR, the packet is always sent to the Y dimension if the link and router in the north direction are non-faulty; otherwise the X dimension is selected. The reason for this selection is that if the distance along the X dimension reaches zero, the source and destination will be located in the same column. Thereby, if there is a fault in one of the components in the remaining path, the packet must take a non-minimal route to bypass the fault. This is not an optimal solution which is addressed by TFLR. TFLR avoids reducing the distance into zero in one direction when the distance along the other direction is greater than one. In other words, when the distance between the current and destination routers reaches one in at least one dimension, at first the possibility of sending the packet through the greater-distance dimension is checked. The packet is sent along the greater-distance dimension if the instant link and router along this direction are non-faulty; otherwise the smaller-distance dimension is examined. Consequently, in Fig. 7b the availabilities of the instant north link and router are checked before than that of the east direction. In the next hop, the packet faces the similar situation as in Fig. 7a, and thus only the shortest paths are selected by TFLR so far. Similarly, in Fig. 7c, where the distances are two hops (or greater than two) and one hop along the X and Y dimensions ($\Delta X \geq 2$ and $\Delta Y = 1$), respectively, the condition of the east direction is examined earlier than that of the north direction. Finally, in Fig. 7d, the east and north direction have the same priority to be selected by packets. By these choices, the packet faces a similar situation as in Fig. 7b or Fig. 7c and thereby only the shortest paths are taken by TFLR in all cases. The idea can be simply extended to the northwest, southeast, and southwest positions in the network.

All possible positions of a single fault are shown in Fig. 8 when a northeastward packet gets close to the destination router. As shown in Fig. 8a when $\Delta X = 1$ and $\Delta Y = 1$, there is a possibility of occurring faults in four links and two routers. Regardless of the location of the fault, the availability of the NE link and N router is checked before the E link and router. In patterns A1, A2, A3, and A6 of Fig. 8a, the NE link and N router are non-faulty and the packet is sent to the north direction, while in patterns A4, A5, and A7, the packet is delivered to the east direction as either the NE link or N router is faulty. In Fig. 8b, when $\Delta X = 1$ and $\Delta Y = 2$, a fault might occur in seven different locations of links and four locations of routers. In all patterns, the availabilities of the N link and the N router are checked before those of the east direction. In patterns B1, B2, B3, B4, B5, B6, B7, B9, B10, and B11, the packet is sent to the

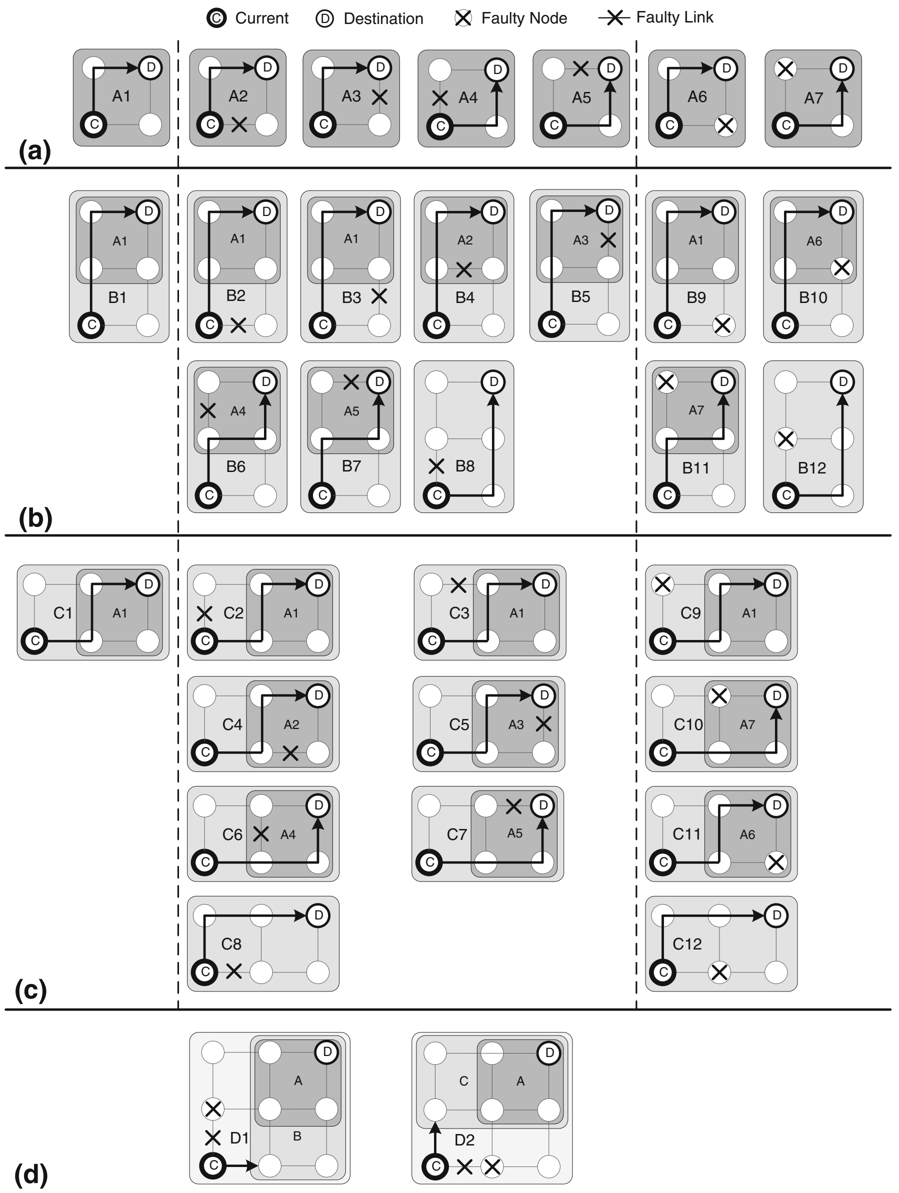

**Fig. 8** All possible positions of a faulty link or router when the packet gets close to the destination router

north direction as both the link and router are non-faulty, while in the next hop, one of the patterns of Fig. 8a arises (i.e. patterns A1, A2, A3, A4, A5, A6, or A7). In patterns B8 and B12 of Fig. 8b, the packet has to be routed first to the east direction and then the north direction in order to reach the destination router. In Fig. 8c, when $\Delta X = 2$ and $\Delta Y = 1$, the availability of the E link and the E router should be examined before the N link and the N router. Therefore, in patterns C1, C2, C3, C4, C5, C6, C7, C9,

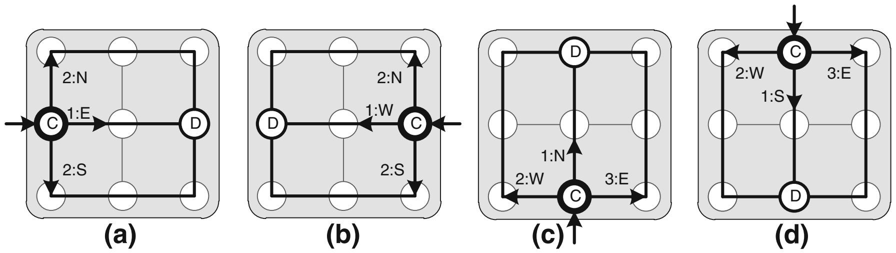

**Fig. 9** Bypassing faults when the destination is located in the **(a)** east **(b)** west **(c)** north, **(d)** south positions of the source router (Note that numbers determine the priority of selecting among different routes)

C10, and C11, the packet is sent to the east direction as both the E link and the E router are non-faulty. In patterns C8 and C12, the packet is delivered to the north direction so that the fault is bypassed. In all the other cases (when $\Delta X \geq 2$ and $\Delta Y \geq 2$ in Fig. 8d, the packet is sent to the non-faulty direction. In the next hop, the patterns are similar to Fig. 8b or Fig. 8c. In sum, all the cases are supported by using only the shortest paths.

As it is already discussed, TFLR is a fully adaptive routing algorithm supporting all the required turns to route packets through the shortest paths. Using TFLR, the northeast, northwest, southeast, and southwest packets do not take any non-minimal routes for tolerating faults.

When the packet is east-, west-, north-, or south- bounded and there is a faulty link or router in the path, the packet must be routed through a non-minimal path around the fault. As illustrated in Fig. 9a, for the eastward packet, at first the east link and router are checked and if they are healthy, the packet is sent through this direction. However, if either the link or the router is faulty, the packet is delivered to the north or south direction. Westward packets do the same behavior (Fig. 9b). For a northward packet facing a fault in the north link or router (Fig. 9c), the west direction is checked earlier than the east direction. It means that rerouting through the east direction is done only when the fault is located in the left borderline. A similar perspective is applied to southward packets (Fig. 9d).

Now, we need to show that all the required turns for bypassing faults are in the set of allowable turns. By investigating the required turns it can be seen that eastward packets use the E-N1, N1-E, E-S1, and S1-E turns to bypass a faulty link or router (Fig. 10a) in which all turns are in the set of allowable turns. Similarly, as shown in Fig. 10b, all the required turns by the westward packets are allowable (i.e. W-N2, N2-W, W-S2, and S2-W).

We should also prove that the northward and southward packets are routed in the network using the allowable turns. As illustrated in Fig. 11a, b, normally the northward and southward packets use the allowable turns as N2-W, W-N2, N2-E, S2-W, W-S2, and S2-E to bypass faults. As shown in Fig. 11c, d, when the source and destination routers are located in the left borderline and there is a faulty link or router in the path, the required turns are N2-E, E-N2, N2-W, S2-E, E-S2, and S2-W. Among them, E-N2 and E-S2 are unallowable according to the turn model of TFLR, but a complete cycle cannot be formed in borderline cases (as indicated in [12]) and these unallowable turns can be safely taken. The

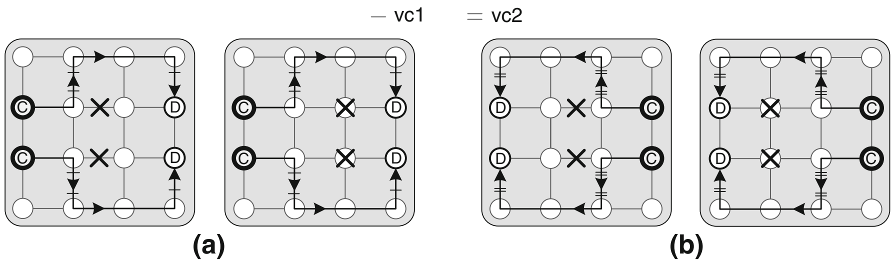

**Fig. 10** Tolerating a faulty router or link by **(a)** eastward packets **(b)** westward packets

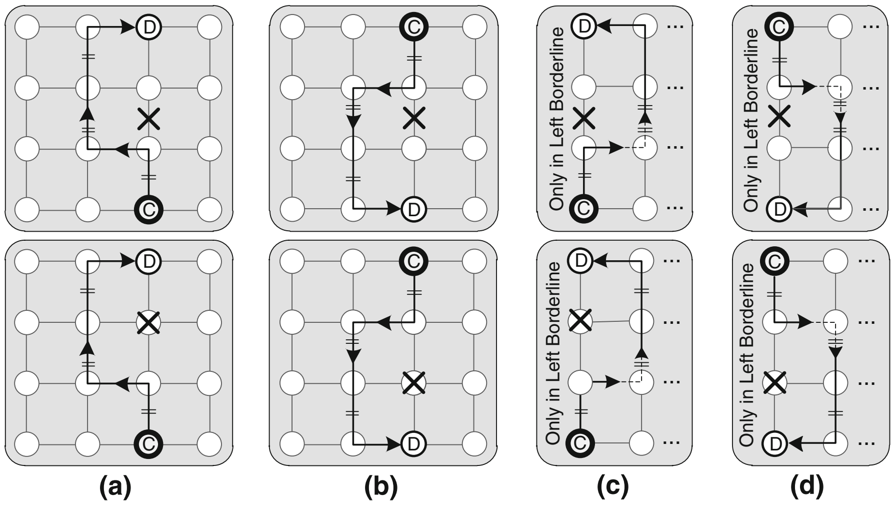

**Fig. 11** Tolerating a fault **(a)** by northward packets in default cases **(b)** by southward packets in default cases **(c)** by northward packets when a fault is in the left borderline **(d)** by southward packets when a fault is in the left borderline

whole TFLR routing algorithm to tolerate both faulty links and routers is shown in Fig. 12.

## 3.3 Distribution of fault information

In all cases, TFLR needs to know at most the statuses of eight surrounding links to tolerate a faulty link (Fig. 13a). Moreover, to tolerate a faulty router, it is enough that each router is informed about the fault statues of its four neighboring routers (Fig. 13b). Each router is aware of the fault in its instant links, but it should be informed about the fault information on the other links and routers as well. Therefore, 3-bit wire is required to transfer the fault information of the north neighboring router and its east and west links to the current router. Similarly, 3-bit wire is needed to transfer the information from the south direction. 1-bit wire is enough to transfer the information of the east and west neighboring routers to the current router. The whole information that is needed by TFLR is shown in Fig. 13c.

```
Definitions:  Xs,Ys,Xd,Yd,Xc,Yc: X and Y coordinates of the source(s), destination(d) and current(c) routers
                                        **-------------------------------------------**
position <= {NE,NW,SE,SW,E,W,N, or S} according to the source and destination positions
x_dir <= E when Xd > Xc else W;
y_dir <= N when Yd > Yc else S;
vc <= vc1 when position={E,NE,SE} else vc2 when position={W,N,S,NW,SW};
Δx <= Xd - Xc when Xd>Xc else Xc-Xd;
Δy <= Yd - Yc when Yd>Yc else Yc-Yd;
if position={NE, NW, SE, or SW} then
    if (Δx>=1 and Δy=0) then select <= x_dir;
    elsif (Δx=0 and Δy>=1) then select <= y_dir(vc);
    elsif (Δx=1 and Δy>=1) then
        if neighbor(y_dir)=faulty or link(y_dir)=faulty then select <= x_dir;
        else select <= y_dir(vc); end if;
    elsif (Δx>=1 and Δy=1) then
        if neighbor(x_dir)=faulty or link(x_dir)=faulty then select <= y_dir(vc);
        else select <= x_dir; end if;
    else
        if neighbor(x_dir)=faulty or link(x_dir)=faulty then select <= y_dir(vc);
        elsif neighbor(y_dir)=faulty or link(x_dir)=faulty then select <= x_dir;
        **else select <= x_dir or y_dir(vc); end if;                 **Adaptive implementation**
        **else select <= x_dir; end if;                              **Deterministic implementation**
    end if;
elsif position={E or W} then
    if Δy=0 then
        if neighbor(x_dir)=faulty or link(x_dir)=faulty then
            **select <= N(vc) or S(vc) based on their availability;       **Adaptive implementation**
            **if (Yc: top-borderline) then select <= S(vc) else N(vc);    **Deterministic implementation**
        else select <= x_dir; end if;
    else
        if neighbor(y_dir)=dest then select <= y_dir(vc);
        else select <= x_dir; end if;
    end if;
elsif position={N or S} then
    if Δx=0 then
        if neighbor(y_dir)=faulty or link(y_dir)=faulty then
            if (Xc: left-borderline) then select <= east; else select <= west; end if;
        else select <= y_dir(vc); end if;
    else
        if neighbor(x_dir)=dest) then select <= x_dir;
        else select <= y_dir(vc); end if;
    end if;
end if;
```

**Fig. 12** The TFLR routing algorithm

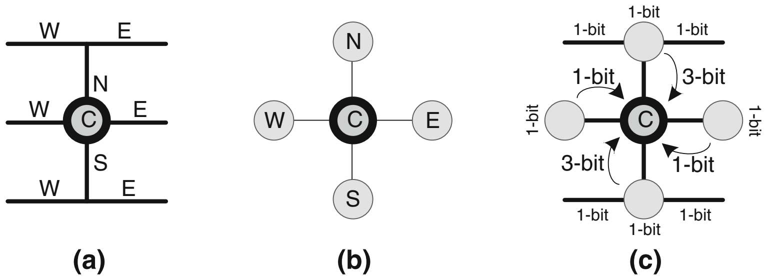

**Fig. 13** The statuses of eight links and four routers are needed by TFLR

## 3.4 Deterministic and adaptive implementation of TFLR

As was already mentioned, in deterministic routing, all packets follow a single path between each pair of source and destination routers. This guarantees that packets reach the destination in the same order in which they have been delivered from the source router and thus no reordering mechanism is needed at destinations. In adaptive routing, however, packets can be routed through different paths toward the destination router which decreases the congestion in the network and improves the performance. Since packets experience different congestion in the network, their arrival orders might be different at the destination router, and thereby a reordering mechanism is needed to handle it. In general, the deterministic routing is more used due to its simplicity while adaptive routing is more preferred for gaining a better performance. TFLR can be implemented in both modes. Fig. 14a shows three examples of deterministic implementation of TFLR when packets are sent from the source routers S1, S2, and S3 toward the destination routers D1, D2, and D3. By default, for example in a fault-free case from S1 to D1, packets are routed in the X dimension until the distance along this dimension reaches one ($\Delta x = 1$). At this point, packets are routed in the Y dimension until the distance reaches zero along this dimension ($\Delta y = 0$). Finally, packets are delivered in the X dimension to reach the destination router ($\Delta x = 0$). Therefore, all packets traverse from S1 to D1 by following the same route, guaranteeing the in-order delivery. In the second example (i.e. S2 to D2), packets are routed in the X dimension until the distance reaches one. Then, packets move along the Y dimension until the distance reaches zero. However, packets face a fault in the path and thus they have to be routed to the X dimension. Finally, they reach the destination by moving along the Y dimension. All the other packets do the same behavior and thus a single path is selected by them. In the third example (i.e. S3 to D3), packets are rerouted to the west direction when facing a fault. Then, they are rerouted along the Y dimension until they reach the same row as the destination router, and finally the X dimension is taken to reach the destination router. Fig. 14b shows an adaptive behavior of FTLR where packets can use almost all the shortest paths between the source and destination.

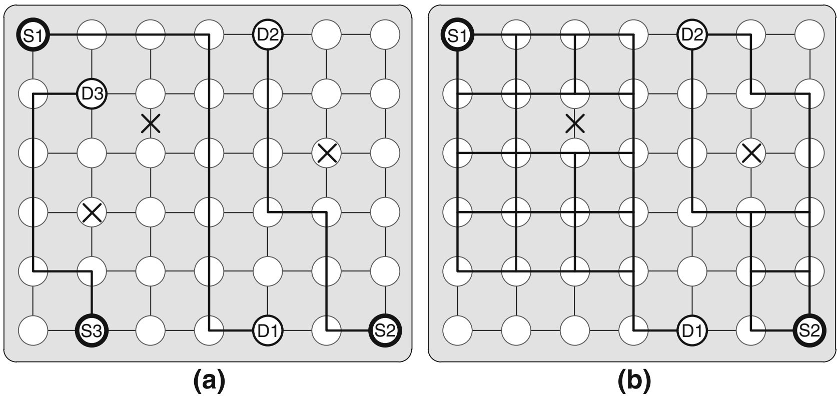

**Fig. 14** **(a)** Deterministic routing **(b)** Adaptive routing

## 4 Results and discussion

To evaluate the efficiency of the proposed fault-tolerant routing scheme, the simulator is developed with VHDL to model all major components of the on-chip network. For all routers, the data width is set to 32 bits and each input channel has 8 buffer slots of 32 bits. Moreover, the packet length is uniformly distributed between 5 and 10 flits. As a performance metric, we use latency defined as the number of cycles between the initiation of a packet by a processing element and the time when the packet is completely delivered to the destination. The simulator is warmed up for 12,000 cycles and then the average performance is measured over another 200,000 cycles. In adaptive routing, when there are alternative choices, a direction with a smaller congestion value is selected. The congestion threshold value is set to 5, meaning that a buffer is considered as a congested one when at least 5 out of 8 buffer slots are occupied.

For evaluating the performance, TFLR, ReRS [12], and RAFT [13] methods are compared together. TFLR is our proposed method which utilizes one and two virtual channels along the X and Y dimension, respectively, and it is able to support either one faulty link or router in all locations of the network. D-TFLR and A-TFLR stand for deterministic and adaptive form of TFLR, respectively. As discussed in the related work, ReRS does not require any virtual channel and it is able to tolerate a single faulty router. RAFT is designed for tolerating faulty links and it is able to tolerate two faulty links in the network in all locations [13]. The RAFT method requires two virtual channels along both dimensions. To have a fair comparison, we utilize two virtual channels for all methods while extra virtual channels are used to improve the performance.

### 4.1 Performance analysis under uniform traffic profile

In the uniform traffic profile, each processing element generates data packets and sends them to another processing element using a uniform distribution [24]. The mesh size is considered 8 × 8. The emphasis of Fig. 15a is on the router's fault so that the average communication latencies of TFLR and ReRS are measured for a fault-free and a single faulty router cases. As observed from the results, in fault-free cases, the ReRS and D-TFLR methods are performing the best as they are based on deterministic

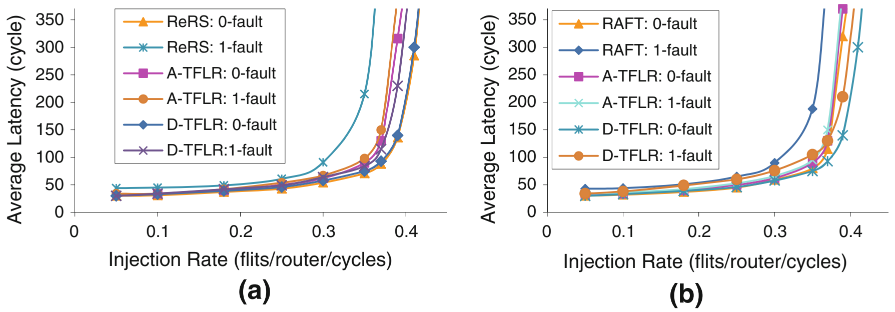

**Fig. 15** Performance analysis in an 8 × 8 mesh network when tolerating **(a)** a faulty router **(b)** a faulty link under the uniform traffic profile

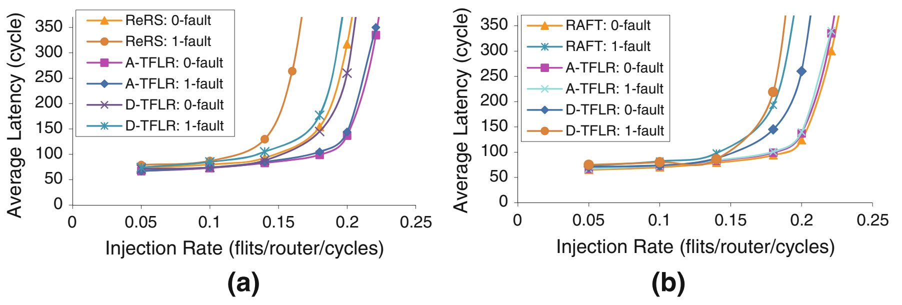

**Fig. 16** Performance analysis in an $8 \times 8$ mesh network when tolerating **(a)** a faulty router **(b)** a faulty link under the hotspot traffic profile

routing (i.e. similar to XY routing) which is well suited to uniform traffic. As shown in this figure, when a single fault occurs in the network, the performance of the ReRS method significantly decreases while the TFLR method maintains the performance in the presence of the fault.

Fig. 15b focuses on the link's fault so that the average communication latencies of TFLR and RAFT are compared in fault-free and a single faulty link cases. The performance of RAFT is slightly better than A-TFLR when there is no fault in the network. The reason is that RAFT is a fully adaptive routing algorithm while in TFLR the adaptivity is limited when packets get close to their destinations. In one-faulty cases, TFLR maintains the performance at a very similar level while the performance of RAFT drops significantly. This is due to the fact that TFLR can route packets through the shortest paths while in RAFT, packets may take longer paths when facing a faulty link.

## 4.2 Performance analysis under hotspot traffic profile

Under the hotspot traffic profile, one or more routers are chosen as hotspot receiving an extra portion of traffic in addition to the regular uniform traffic. In simulations, given a hotspot percentage of $H$, a newly generated packet is directed to each hotspot router with an additional $H$ percent probability. We simulate the hotspot traffic with a single hotspot router with $H = 10\,\%$ at (4,4) in an $8 \times 8$ mesh network. In Fig. 16a, the performance of TFLR and ReRS is measured for fault-free and one-faulty router cases while in Fig. 16b, the performance of TFLR and RAFT is measured for fault-free and one-faulty link cases. In all configurations and in both figures, A-TFLR outperforms the other approaches. This is due to the fact that A-TFLR not only avoids unnecessary longer paths but also reduces the congestion around the faulty region, resulting in a better performance.

## 4.3 Reliability evaluation under uniform traffic profile

For measuring reliability, we take the adaptive version of TFLR into consideration. The reliability of TFLR is compared with ReRS and RAFT in terms of faulty routers or links, respectively. We increase the number of faults from 1 to 6. All faults are

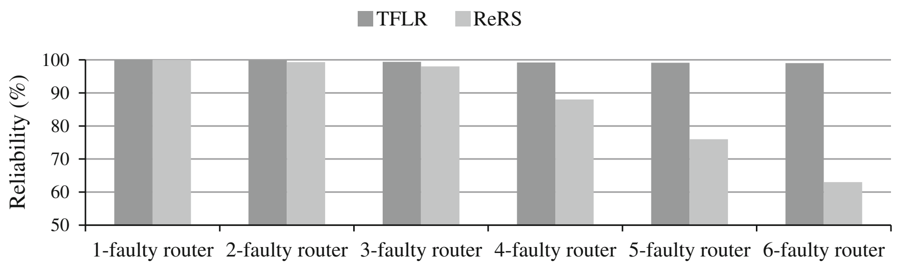

**Fig. 17** Reliability evaluation of TFLR and ReRS in a 6 × 6 mesh network under uniform traffic profile

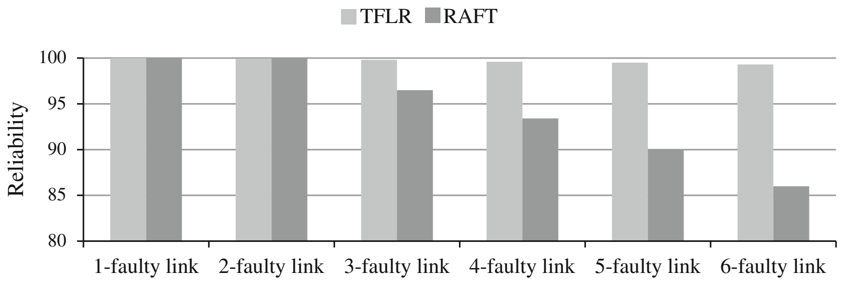

**Fig. 18** Reliability evaluation of TFLR and RAFT in a 6 × 6 mesh network under uniform traffic profile

selected using a random function. The results are obtained using 10,000 iterations in a 6 × 6 mesh network when the underlying traffic is uniform random. The network is counted as reliable if all the injected packets reach their destinations. As shown in Figs. 17 and 18, TFLR can tolerate up to 6 faulty routers and links by more than 99 % reliability.

## 4.4 Hardware analysis

In addition to performance, area overhead and power consumption play an important role in the efficiency of many-core architecture [25,26]. To assess the area overhead and power consumption of the on-chip implementation, the whole platform of each method is synthesized using Synopsys Design Compiler. We measured the area overhead and power consumption of the TFLR, ReRS, and RAFT methods. RAFT uses two virtual channels in both X and Y dimensions. TFLR utilizes one and two virtual channels along the X and Y dimensions, respectively, while ReRS does not use any virtual channel. The power consumption is measured under a single faulty link for RAFT and a single faulty router for TFLR and ReRS. Each scheme includes network interfaces, routers, and communication channels. All methods are synthesized using the TSMC 65nm technology at the operating frequency of 500 MHz and supply voltage of 1V. We perform place-and-route, using Cadence Encounter, to have precise power and area estimations. The power dissipation is calculated using Synopsys PrimePower in an 8 × 8 mesh network. The layout area and power consumption of each platform are shown in Table 2. As indicated in this table, RAFT has a larger area overhead than TFLR and ReRS due to using more virtual channels. The area overhead of TFLR is

**Table 2** Details of hardware implementation

| Network platforms | Area (mm$^2$) | Power (mw) |
|---|---|---|
| TFLR | 2.575 | 1.567 |
| ReRS | 2.126 | 1.466 |
| RAFT | 2.927 | 1.791 |

larger than ReRS as TFLR utilizes one virtual channel more than ReRS. The power consumption of TFLR is slightly larger than ReRS. It is because of using the shortest paths and avoiding hotspots in the network in the presence of faults.

## 5 Summary and conclusion

In this paper, we proposed a fault-tolerant routing algorithm, called TFLR. This algorithm is general and covers both faulty links and routers in the network. It guarantees to tolerate all locations of a single fault in the network. TFLR is not only a fault-tolerant method but also makes attention to the performance as one of its main priorities. It is able to maintain the performance of the network in the presence of faults because packets are routed through the shortest paths between each pair of source and destination routers as long as a path exists. TFLR can be configured in both adaptive and deterministic modes. It can be employed in the deterministic mode if an in-order delivery is needed while for achieving a better performance it can be implemented in the adaptive mode, but an in-order delivery mechanism would be needed. Finally, the TFLR routing algorithm is very simple with a negligible area overhead.

## References

1. Palesi M, Daneshtalab M (2014) Routing Algorithms in Networks-on-Chip. Springer, Berlin
2. Ni LM, McKinley PK (1993) A survey of wormhole routing techniques in direct networks. Computer 26(2):62–76
3. Daneshtalab M, Ebrahimi M, Liljeberg P, Plosila J, Tenhunen H (2010) A low-latency and memory-efficient On-chip network. In: proceedings of the Fourth ACM/IEEE international symposium on networks-on-Chip (NOCS), pp 99–106
4. Ebrahimi M, Daneshtalab M, Liljeberg P, Plosila J, Tenhunen H (2012) “CATRA- congestion aware trapezoid-based routing algorithm for on-chip networks”. In: Proceedings of the design, automation test in Europe conference exhibition (DATE), pp 320–325
5. Ebrahimi M, Daneshtalab M, Liljeberg P, Plosila J, Tenhunen H (2012) “LEAR - A low-weight and highly adaptive routing method for distributing congestions in On-chip networks”, In: Proceedings of 20th euromicro international conference on parallel, distributed and network-based processing (PDP), pp 520–524
6. Wang X, Mak T, Yingtao Jiang MY, Daneshtalab M, Palesi M (2013) On self-tuning networks-on-chip for dynamic network-flow dominance adaptation. In:proceedings of seventh IEEE/ACM international symposium on networks on chip (NoCS), pp 1–8
7. Daneshtalab M, Ebrahimi M, Liljeberg P, Plosila J, Tenhunen H (2013) A systematic reordering mechanism for on-chip networks using efficient congestion-aware method. J Syst Archit (JSA-elsevier), 59(4–5):213–222
8. Ebrahimi M, Daneshtalab M, Sreejesh NP, Liljeberg P, Tenhunen H (2009) Efficient network interface architecture for network-on-chips. In: proceedings of NORCHIP, pp 1–4
9. Fick D, DeOrio A, Hu J, Bertacco V, Blaauw D, Sylvester D (2009) Vicis: A reliable network for unreliable silicon. In: 46th ACM/IEEE design automation conference (DAC), pp 812–817

10. Boppana RV, Chalasani S (1995) Fault-tolerant wormhole routing algorithms for mesh networks. IEEE Trans Comput 44(7):848–864
11. Sui P-H, Wang S-D (1997) An improved algorithm for fault-tolerant wormhole routing in meshes. IEEE Trans Comput 46(9):1040–1042
12. Zhang Z, Greiner A, Taktak S (2008) A reconfigurable routing algorithm for a fault-tolerant 2D-mesh network-on-Chip. In: proceedings of 45th ACM/IEEE design automation conference (DAC), pp 441–446
13. Valinataj M, Mohammadi S, Plosila J, Liljeberg P, Tenhunen H (2011) A reconfigurable and adaptive routing method for fault-tolerant mesh-based networks-on-chip. AEU - Int J Electron Commun 65(7):630–640
14. Wu J (2003) A fault-tolerant and deadlock-free routing protocol in 2D meshes based on odd–even turn model. IEEE Trans Comput 52(9):1154–1169
15. Chiu G-M (2000) The odd-even turn model for adaptive routing. In: IEEE transactions on parallel and distributed systems 11(7):729–738
16. Tsai W-C, Zheng D-Y, Chen S-J, Hu Y-H (2011) A fault-tolerant NoC scheme using bidirectional channel. In: proceedings of 48th ACM/EDAC/IEEE design automation conference (DAC), pp 918–923
17. Koibuchi M, Matsutani H, Amano H, Mark Pinkston T (2008) A Lightweight Fault-Tolerant Mechanism for Network-on-Chip. In: proceedings of the second ACM/IEEE international symposium on networks-on-Chip (NoCS), pp 13–22
18. Ebrahimi M, Daneshtalab M, Plosila J, Tenhunen H (2012) MAFA: adaptive fault-tolerant routing algorithm for networks-on-chip", In: proceedings of 15th euromicro conference on digital system design (DSD), pp 201–207
19. Fick D, DeOrio A, Chen G, Bertacco V, Sylvester D, Blaauw D (2009) A highly resilient routing algorithm for fault-tolerant NoCs. In: proceedings of design, automation test in Europe conference exhibition (DATE), pp 21–26
20. Ebrahimi M, Daneshtalab M, Plosila J, Farhad Mehdipour (2013) MD: minimal path-based fault-tolerant routing in on-chip networks", In: proceedings of the IEEE/ACM 18th Asia and South Pacific design automation conference (ASP-DAC), pp 35–40
21. Ebrahimi M, Daneshtalab M, Plosila J, Tenhunen H (2013) Minimal-path fault-tolerant approach using connection-retaining structure in networks-on-chip", In: proceedings of 7th international symposium on networks-on-Chip (NOCS), pp 1–8
22. Ebrahimi M, Daneshtalab M, Plosila J (2013) High performance fault-tolerant routing algorithm for NoC-based many-core systems", In: proceedings of 21th IEEE euromicro conference on Parallel, Distributed and network-based computing (PDP), pp 463–469
23. Glass CJ, Glass CJ, Ni LM, Ni LM (1992) Maximally fully adaptive routing in 2D Meshes. In: proceedings of international conference on parallel processing, pp 101–104
24. Glass CJ, Ni LM (1992) "The Turn Model for Adaptive Routing". In proceedings of the 19th, annual international symposium on computer architecture pp 278–287
25. Wang ZL, Yang M, Jiang Y, Daneshtalab M, Mak T (2013) A low cost, high performance dynamic-programming-based adaptive power allocation scheme for many-core architectures in the dark silicon Era. In: proceedings of the 11th IEEE symposium on embedded systems for real-time multimedia (ESTImedia)
26. Carabano J, Dios F, Daneshtalab M, Ebrahimi M (2013) An exploration of heterogeneous systems. In: 2013 8th international workshop on reconfigurable and communication-centric Systems-on-Chip (ReCoSoC), pp 1–7