High Performance Fault-Tolerant Routing Algorithm for NoC-based Many-Core Systems

Masoumeh Ebrahimi, Masoud Daneshtalab, Juha Plosila
Department of Information Technology, University of Turku
{masebr, masdan, juplos}@utu.fi

Abstract— **Networks-on-Chip (NoCs) has become a promising approach for the on-chip communication infrastructure of many-core Systems-on-Chip (SoCs). Faults may occur in the NoC both at the router and link level. There are many fault-tolerant approaches presented both in the off-chip and on-chip networks. Some approaches disable some healthy components in order to form a specific shape and others not. Regardless of all varieties, there has always been a common assumption among them. Most of all traditional fault-tolerant methods are based on rerouting packets around a faulty node or region. These approaches affect the performance significantly not only by taking longer paths but also by creating hotspot around a fault. The focus of this paper is to maintain the performance of NoC in the presence of faults. The presented method takes advantage of a fully adaptive routing algorithm using one and two virtual channels along the X and Y dimensions. This method is able to tolerate all cases of one-faulty node without losing the performance of NoC. According to the experimental results, this presented fault-tolerant routing algorithm is able to support up to six faulty nodes in the 8×8 mesh network by up to 98% reliability.**

Keywords-component: Networks-on-Chip; Fully adaptive algorithms; fault-tolerant approaches; the shortest paths

I. INTRODUCTION

As the number of processing elements integrated in a single chip is increasing, traditional bus-based architectures in Many-Core Systems-on-Chip (MCSoCs) are inefficient and new communication infrastructure is needed. Networks-on-Chip (NoC) has emerged as a promising solution for on-chip interconnection in MCSoCs due to its scalability, reusability, flexibility, and parallelism [1]-[4].

Transient and permanent faults are two different types of faults that can occur in on-chip networks [5][6]. Transient faults are temporary and unpredictable. They are often difficult to be detected and corrected. Permanent faults are caused by physical damages such as manufacturing defects and device wear-out. In this paper, permanent faults are taken into consideration.

Routing techniques provide some degrees of fault tolerance in NoCs which can be categorized into deterministic and adaptive [7]-[12]. A deterministic routing algorithm uses a fixed path for each pair of nodes resulting in increased packet latency especially in congested networks. Implementations of deterministic routing algorithms are simple but they are unable to balance the load across the links in a non-uniform traffic. In adaptive routing algorithms, a packet is not restricted to a single path when traveling from a source node to its destination. So they can decrease the probability of routing packets through congested or faulty regions. That is, adaptive routing algorithms not only can avoid congestion in the network but also can provide better fault-tolerant characteristics by utilizing alternative routing paths.

Virtual channels can be used for different purposes in the network: avoiding deadlock, increasing performance, and tolerating faults, but they are imposing extra costs.

In this paper, we present a novel routing algorithm named **Hi**gh **P**erformance **Fa**ult-tolerant **R**outing (HiPFaR). The main idea of this algorithm is to tolerate faults without affecting the performance. We take a fully adaptive routing algorithm into account with the minimum number of virtual channels (i.e. one and two virtual channels along the X and Y dimensions). There are eight different positions regarding the source and destination nodes as east, west, north, south, northeast, northwest, southeast, and southwest. In this category, east-, west-, north-, and south-ward packets have to take a non-minimal path when facing faults. To support non-minimal routing, a set of allowable turns is defined in the network. It is proven that the network using these turns is deadlock free. Moreover, it is shown that faulty nodes in the network can be tolerated by northeast-, northwest-, southeast-, and southwest-ward packets without taking non-minimal paths. The proposed algorithm needs only the fault information of four direct neighboring nodes in order to make a correct decision.

The rest of this paper is organized as follows: Section II reviews the related work. Preliminaries are given in Section III. The proposed fault-tolerant routing algorithm is presented in section IV. The results are investigated in Section V while we summarize and conclude in the last section.

II. RELATED WORK

In interconnection networks, there are two different groups of fault-tolerant routing algorithms dealing with permanent faults. The first group, handling convex shapes, is based on defining fault ring or fault chain around faulty regions. Some healthy nodes might be disabled in order to form a specific shape [13][14][15][16].

The second group utilizes the contour strategy for addressing faults [17][18][19] which is divided in two subgroups: methods using virtual channels [18][20][21] and those without using virtual channels [19][22]. The virtual channel-based fault-tolerant routing algorithms are more efficient than those without virtual channels.

In [17], the deterministic routing is able to tolerate all one-faulty routers in 2D mesh network without using virtual channels and disabling healthy nodes. In this algorithm, based on the convex areas, a reconfigurable routing algorithm is used

to provide the possibility of routing packets through a cycle free surrounding the convex areas. However, to support more faulty nodes, the contours must not be overlapped and thus faulty routers should be located far away from each other. In addition, this method requires collecting the fault information of at least eight neighboring nodes (direct and indirect).

In this paper, the proposed fault-tolerant routing algorithm is based on a fully adaptive routing algorithm using the minimum number of virtual channels. It requires collecting the fault information of only four neighboring nodes. Based on this knowledge, it is able to deliver packets through the available shortest paths.

III. PRELIMINARIES

Fig. 1(a) shows a typical router in the XY network. In this figure, each input channel is paired with a corresponding output channel. By adding two virtual channels per physical channel, a double-XY network is obtained (Fig. 1(b)). The virtual channels in each dimension are differentiated by vc1 and vc2. Fig. 1(c) shows the double-Y network in which one and two virtual channels are used along the X and Y dimensions, respectively. Each router in the double-Y network has seven pairs of channels, i.e. East(E), West(W), North-vc1(N1), North-vc2(N2), South-vc1(S1), South-vc2(S2), and Local(L). The idea of this paper is developed upon a double-Y network. However it can be implemented on a double-XY network or a network with a larger number of virtual channels.

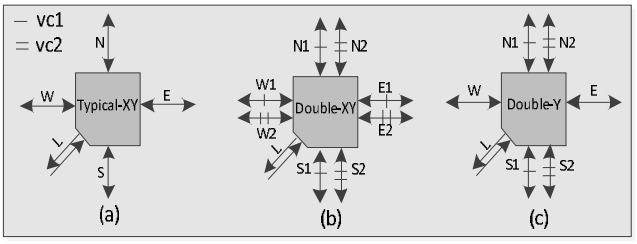

Fig. 1. A router in (a) XY (b) double-XY (c) double-Y network

IV. MINIMAL PATH ROUTING IN FAULTY NETWORK

In this section, we present a routing algorithm named High Performance Fault-tolerant Routing (HiPFaR). Unlike traditional methods, by using HiPFaR, packets possibly have alternative choices of minimal paths when facing a faulty node. By this approach, all packets can be routed through the network using the shortest paths, as long as a path exists. We start by defining a set of allowable turns in which non-minimal paths can be used to support faults in the case of east-, west-, north-, and south-ward packets. Then, it is proven that thenetwork using these allowable turns is deadlock free. We continue by examining the possibility of choosing the shortest paths for northeast-, northwest-, southeast-, and southwest-ward packets.

A. Turn Models and Tolerating Faults

The proposed fault-tolerant routing algorithm is based on fully adaptive routing algorithms. In this paper, one and two virtual channels are used along the X and Y dimensions. This is the minimum number of virtual channels that can be employed to provide fully adaptiveness. In double-Y networks, commonly the following method is used to guarantee the deadlock freeness. The network is partitioned into two sub-networks called +X and –X, each having half of the channels in the Y dimension. Eastward packets are routed through +X sub-network (i.e. by using the first virtual channel (vc1) in the Y dimension) while westward packets are propagated within -X sub-network (i.e. by using the second virtual channel (vc2) along the Y dimension). Allowable and non-allowable turns are shown in Fig. 2(a). According to this figure the turns N1-E, E-S1, S1-E, E-N1, W-N2, S2-W, W-S2, N2-W, S1-S1, N1-N1, S2-S2, and N2-N2 are allowable while the others are prohibited.

One of the aims of this paper is to tolerate faults using the available shortest paths. A non-minimal route is necessitated when the source and destination nodes are located in the same row or column with a faulty node between them. Fig. 3(a) and Fig. 3(b) indicate the cases where the destination of a packet is to the east and west of the source, respectively, and there is a faulty node in the path. In these cases, there are two options to misroute a packet when facing a faulty node: turning to the north direction (i.e. the given turns are E-N1, N1-E, and E-S1 in Fig. 3(a) and W-N2, N2-W, and W-S2 in Fig. 3(b)) or turning to the south direction (i.e. the given turns are E-S1, S1-E, and E-N1 in Fig. 3(a) and W-S2, S2-W, and W-N2 in Fig. 3(b)). By investigating the required turns to perform misrouting, it can be obtained that all of the turns are within the set of allowable turns (Fig. 2(a)). Therefore, east- and west-ward packets can turn either to the north or south direction when facing faults. The situation is different for north- and south-ward packets since all the required turns are not in a set of allowable turns. For example, in Fig. 3(c) and Fig. 3(d), the turns N2-E, N1-W, S2-E, and S1-W are prohibited. One might think that alternative turns like N1-E, N2-W, S1-E, and S2-W are allowable and can be taken. However, according to Fig. 2(a) packets cannot switch from the first to the second virtual channel along the Y dimension or vice versa (i.e. the turns N2-N1, N1-N2, S1-S2, and S2-S1 are not allowable in Fig. 2(a)).

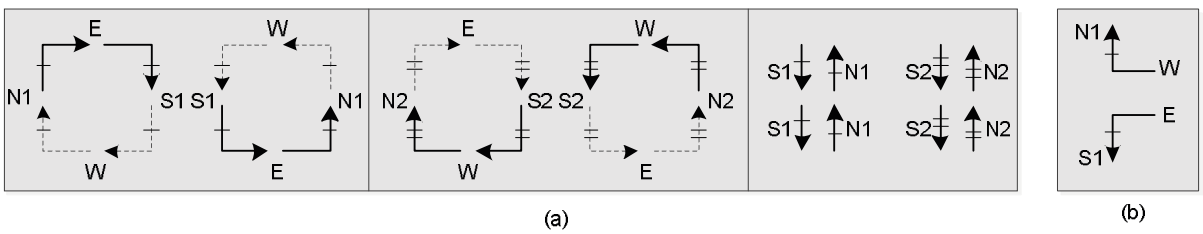

Fig. 2. Allowable and non-allowable turns in the double-Y network

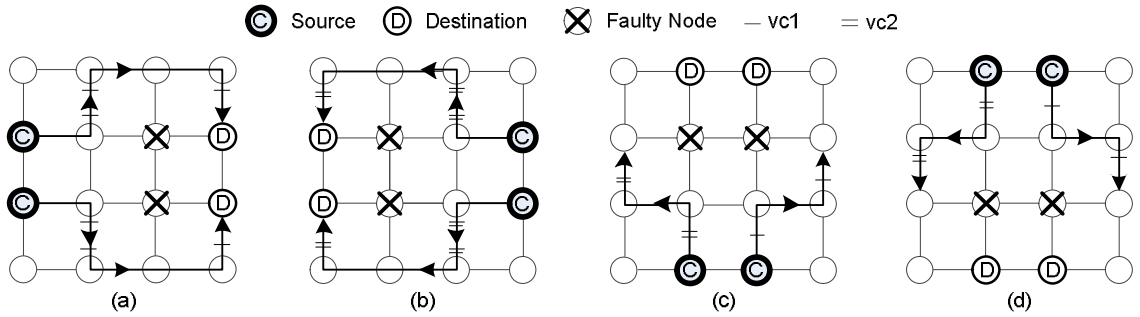

Fig. 3. The possibility of tolerating faulty nodes using the allowable turns of Fig. 2(a) when the destination is in the (a) east (b) west (c) north (d) south direction of the source node

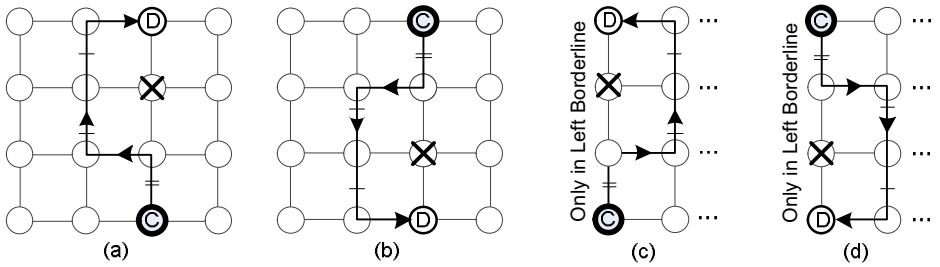

Fig. 4. Tolerating faulty nodes by using non-minimal paths when the destination is in the (a) north (b) south direction. The exceptional cases are shown in (c) and (d) where a fault occurs on the left borderline

To allow north- and south-ward packets to turn around faults, two more turns are added into the set of allowable turns. These turns are W-N1 and W-S1, indicated in Fig. 2(b). In this way, north- and south-ward packets should be misrouted to the west direction when facing faults (Fig. 4(a) and Fig. 4(b)). Exceptionally, when the fault occurs in the left borderline, packets have to take the east direction to bypass the fault. This results in taking N1-W and S1-W turns which are prohibited by HiPFaR. However, according to [17] a cycle cannot be completed in borderlines and thus these non-allowable turns can be safely taken in these cases.

B. HiPFaR is Deadlock Free

It should be proven that by allowing two additional turns in HiPFaR, the network remains deadlock free. We use a numbering mechanism similar to the mad-y routing algorithm [23]. A two-digit number (a, b) is assigned to each output channel of a switch in n×m mesh network. According to the numbering mechanism, a turn connecting the input channel (a_{ic}, b_{ic}) to the output channel (a_{oc}, b_{oc}) is called an ascending turn when (a_{oc}>a_{ic}) or ((a_{oc}=a_{ic}) and (b_{oc}>b_{ic})). Fig. 5 shows how the channels of a switch at the position (x,y) are numbered. By using this numbering mechanism, it is guaranteed that all allowable turns in Fig. 2(a) and Fig. 2(b) are taken in the strictly increasing order, so that the HiPFaR routing algorithm is deadlock free. For instance, if the W-N1 turn is taken into consideration, the west input channel with label (a_{ic}= m+x, b_{ic}=0) is connected to the first virtual channel of the north output port having the label (a_{oc}=m+x, b_{oc}=1+y). This turn takes place in an ascending order since ((a_{oc}=a_{ic}) and (b_{oc}>b_{ic})). Similarly, all the other turns allowed by HiPFaR are taken in an ascending order.

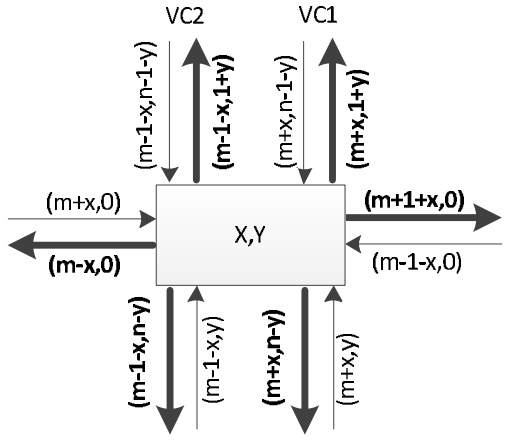

Fig. 5. The numbering mechanism of HiPFaR, similar to mad-y [23]

C. Shortest Paths and Tolerating Faulty Nodes

Let us assume that there is a faulty node in the network. As it is already discussed, the fault is bypassed using non-minimal paths when the source and destination nodes are located in the same row or column. In other cases, only the shortest paths are taken from the source to the destination node. In Fig. 6, for the ease of understanding we investigate HiPFaR for a northeast-ward packet. However, as it will be shown in Subsection IV-D, the HiPFaR routing algorithm is general and can be applied to northwest-, southeast-, and southwest-ward packets without any implementation differences.

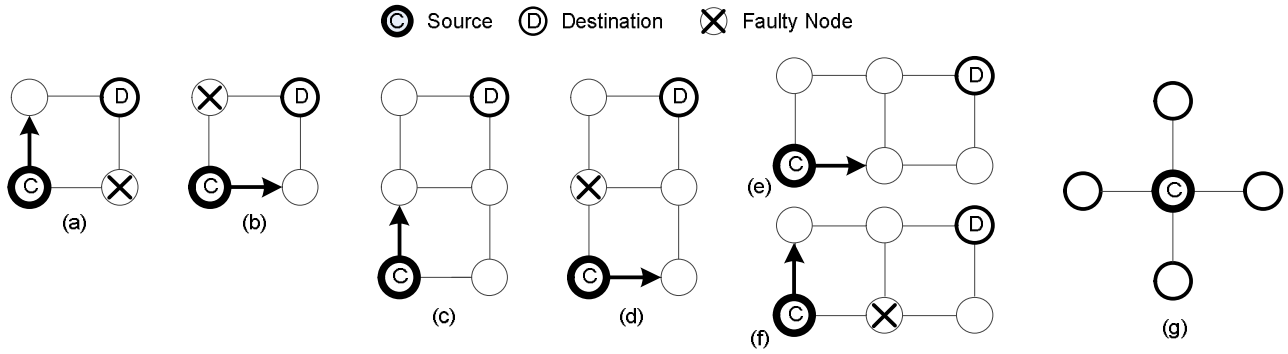

Fig. 6. The basic rules for selecting among the neighboring nodes when a packet gets close to the destination node

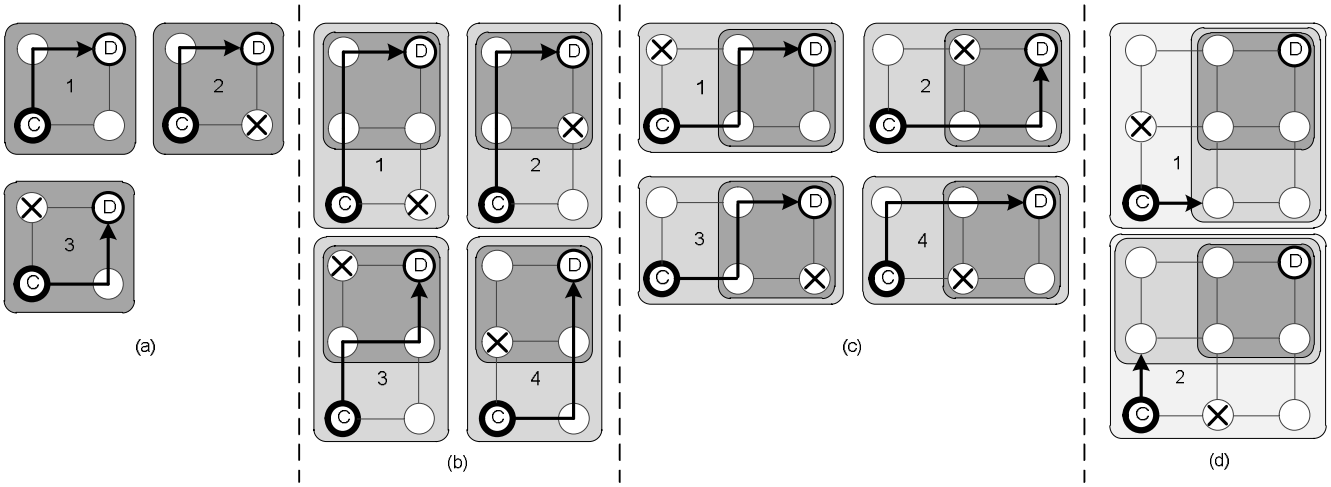

Fig. 7. Different positions of current, destination and a faulty node

As shown in Fig. 6(a) and Fig. 6(b), the packet is one hop away from the destination node in both the X and Y dimensions (*x-dir=1* and *y-dir=1*). By default, the packet is sent to the Y direction (Fig. 6(a)). However, when the north neighboring node is faulty, the packet is delivered to the X direction (Fig. 6(b)). Fig. 7(a) shows the possible cases where the distances along both the X and Y directions are one hop. As illustrated in this figure, in positions 1 and 2, the packet is delivered to the Y direction since the north neighboring node is healthy. In positions 3, the packet is sent to direction X since the north neighboring node is faulty. Thereby, in all three positions, the packet can reach the destination using the shortest paths.

In Fig. 6(c) and Fig. 6(d), the distance is one and two (or greater than two) hops along the X and Y directions (*x-dir=1* and *y-dir>=2*), respectively. The rule is similar to the previous case in which the packet is sent to the Y direction unless the north neighboring node is faulty (i.e. in this case, the packet is delivered to the X direction). According this rule, in positions 1, 2, and 3 of Fig. 7(b), since the north neighboring node is healthy, the packet is sent to the Y direction. In the next hop, the packet stands in one of the positions of Fig. 7(a) (i.e. already shown that packets can reach destination node using the shortest paths in the presence of fault). In position 4, the north neighboring node is faulty, and thus the packet is sent to the X direction. This packet reaches the destination using the shortest path as the faulty node is already bypassed.

Fig. 6(e) and Fig. 6(f) indicate the cases where the distances are two hops (or greater than two) and one hop (*x-dir>=2* and *y-dir=1*) along the X and Y dimensions. The rule is as simple as avoiding to send the packet in the Y direction when the east neighboring node is healthy. Fig. 7(c) shows the different positions of the current, destination and a faulty node. In positions 1, 2, and 3, the packet is sent to the X direction as the east neighboring node is healthy. In the next hop, the packet stands in one of the positions of Fig. 7 (a). If the east neighboring node is faulty (position 4), the packet is delivered to the destination through the north neighboring node.

When the distances along both directions are two or greater than two hops, the packet is sent to a non-faulty neighboring node (Fig. 7(d)). By routing the packet with this policy, the packet reaches one of the positions of Fig. 7(b) or Fig. 7(c). In sum, in all faulty cases, the packet is routed to the destination node through the shortest paths. In a case where both neighboring nodes are healthy, one direction is chosen based on the congestion information. Therefore, HiPFaR is fully adaptive as long as the remaining distance along both directions is equal or greater than two hops.

According to these rules, each node needs to know only the fault information of the four neighboring nodes shown in Fig. 6(g).

D. The HiPFaR Routing Algoritm

The HiPFaR routing algorithm can be explained into two parts, when the destination is in the east, west, north and south directions of the source node (Fig. 8); and when the destination is in the northeast, northwest, southeast, and southwest directions of the source node (Fig. 9). In these figures, the position is specified based on the source and destination position. The *x-dir* and *y-dir* parameters determine the minimal directions toward the destination node regarding the current node. The *delta-x* and *delta-y* parameters maintain the remaining distances from the current to the destination node along the X and Y dimensions. Finally, *vc* indicates the proper virtual channel. According to HiPFaR, the first virtual channel of the Y dimension is used when the destination is toward the east, northeast, and southeast directions of the source node. Similarly, the second virtual channel is utilized for west-, northwest-, and southwest-ward packets. The north- and south-ward packets start routing in the second virtual channel. They switch to the first virtual channel after bypassing a fault. In Fig. 8 when the packets are either east- or west-ward and *delta-y* is equal to zero, there are two options: making a turn to either the north or south direction when and there is a fault in the path; otherwise continuing in the *x-dir* direction. However, when the packet is already misrouted to the north or south direction, it has to be routed along the *x-dir* until *delta-x* becomes zero. At this point, the packet can be delivered to the destination node by turning to the Y direction. The north- and south-ward packets start routing in the second virtual channel as long as they do not face a fault. To bypass the fault, normally the turn to the west direction is made. In a case where the faulty node is located in the left borderline, the turn to the east direction should be made. In the next hop, the packet switches to the first virtual channel of the Y direction and continue routing along this direction until *delta-y* becomes zero. At this point, a turn to the east or west direction is needed to deliver the packet to the destination node.

A minimal routing algorithm is used for northeast-, northwest-, southeast-, and southwest-ward packets. Packets can be adaptively routed within the network when they are far away from the destination node (two or more than two hops along both X and Y dimensions). At each intermediate node, the congestion information is used to send a packet to one of the non-faulty neighboring nodes which are located in the minimal path. When the distance along the X direction reaches one, the packet should be always sent to the Y direction. The exception case is that the north neighboring node is faulty and thus the packet should be delivered to the X direction. On the other hand, when the distance along the Y direction is one while the distance along the X direction is greater than one, the packet should be sent to the X direction unless the east neighboring node is faulty. When the distance reaches zero along the X or Y direction, the packet should be sent through the non-zero direction. All the intermediate nodes in the remaining path would be healthy (by assuming one faulty node in the network,) as the fault is already bypassed. If there are more faulty nodes in the network, in most cases the packet bypasses the faults before getting close to the destination. However, there are some cases in which non-minimal paths are required in order to bypass multiple faulty nodes. Since the scope of this paper is only minimal paths, we simply assume that these cases of multiple faulty nodes could not be supported by HiPFaR.

```
Dx,Dy: X and Y positions of the destination node
Sx,Sy: X and Y positions of the source node
Cx,Cy: X and Y positions of the current node
E: East; W: West;
N: North; S: South;
vc: Virtual Channel;
------------------------------
position <= E WHEN Dx>Sx AND Dy=Sy;
position <= W WHEN Dx<Sx AND Dy=Sy;
position <= N WHEN Dx=Sx AND Dy>Sy;
position <= S WHEN Dx=Sx AND Dy<Sy;

x_dir <= E if Dx>Cx ELSE W;
y_dir <= N if Dy>Cy ELSE S;

vc <= vc1 WHEN position={E} ELSE
      vc2 WHEN position={W};

delta_x <= Dx-Cx when Dx>Cx else Cx-Dx;
delta_y <= Dy-Cy when Dy>Cy else Cy-Dy;
------------------------------
IF position={E or W} THEN
   IF (delta_Y=0) THEN
        IF neighboring_node(x_dir)=faulty THEN
           choice <= N(vc) or S(vc);
        ELSE
           choice <= x_dir;
        END IF;
   ELSE
        IF neighboring_node(y_dir)=dest. THEN
           choice <= y_dir(vc);
        ELSE
           choice <= x_dir;
        END IF;
   END IF;
ELSIF position={N or S} THEN
   IF (delta_x=0) THEN
        IF neighboring_node(y_dir)=faulty THEN
           IF Cx/=0 THEN
                choice <= west direction;
           ELSE  --left borderline
                choice <= east direction;
           END IF;
        ELSE
           choice <= y_dir(vc2);
        END IF;
   ELSE
        IF neighboring_node(x_dir)=dest. THEN
           Choice <= x_dir;
        ELSE
           Choice <= y_dir(vc1);
        END IF;
   END IF;
END IF;
```

Fig. 8. Pseudo VHDL code of HiPFaR when destination is in the east, west, north, or south direction of the source node

```
Dx,Dy: X and Y positions of the destination node
Sx,Sy: X and Y positions of the source node
Cx,Cy: X and Y positions of the current node
E: East; W: West;
N: North; S: South;
vc: Virtual Channel;
-----------------------------
position <= NE WHEN Dx>Sx AND Dy>Sy;
position <= NW WHEN Dx<Sx AND Dy>Sy;
position <= SE WHEN Dx>Sx AND Dy<Sy;
position <= SW WHEN Dx<Sx AND Dy<Sy;

x_dir <= E WHEN Dx>Cx ELSE W;
y_dir <= N WHEN Dy>Cy ELSE S;

vc <= vc1 WHEN position={NE or SE} ELSE
      vc2 WHEN position={NW or SW};

Delta_x <= Dx-Cx WHEN Dx>Cx else Cx-Dx;
Delta_y <= Dy-Cy WHEN Dy>Cy else Cy-Dy;
-----------------------------
IF position={NE, NW, SE, or SW} THEN
   IF (delta_x>=1 AND delta_y=0) THEN
        choice <= x_dir;
   ELSIF (delta_x=0 AND delta_y>=1) THEN
        choice <= y_dir(vc);
   ELSIF (delta_x=1 AND delta_y>=1) THEN
       IF neighboring_node(y_dir)=faulty THEN
          choice <= x_dir;
        ELSE
          choice <= y_dir(vc);
        END IF;
   ELSIF (delta_x>1 AND delta_y=1) THEN
       IF neighboring_node(x_dir)=faulty THEN
          choice <= y_dir(vc);
        ELSE
          choice <= x_dir;
         END IF;
   ELSIF (delta_x>=2 AND delta_y>=2) THEN
        IF neighboring_node(x_dir)=faulty THEN
          choice <= y_dir(vc);
        ELSIF neighboring_node(y_dir)=faulty THEN
          choice <= x_dir;
        ELSE
          choice <= x_dir or y_dir(vc);
         END IF;
   END IF;
END IF;
```

Fig. 9. Pseudo VHDL code of HiPFaR when destination is in northeast, northwest, southeast, or southwest direction of the source

V. EXPERIMENTAL RESULTS

To evaluate the efficiency of the proposed routing scheme, a NoC simulator is developed with VHDL to model all major components of the on-chip network [24][25]. For all the routers, the data width is set to 32 bits. Each input buffer can accommodate 8 flits in each virtual channel. 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 message issued by a Processing Element (PE) and the time when the message is completely delivered to the destination PE. The request rate is defined as the ratio of the successful message injections into the network over the total number of injection attempts. The simulator is warmed up for 12,000 cycles and then the average performance is measured over another 200,000 cycles.

We defined our baseline as a detour strategy similar to [17]. To have a fair comparison, one and two virtual channels are used along the X and Y directions, respectively, for the baseline model. A virtual channel is selected based on the free number of buffer slots at the input buffer of the neighboring routers. Unlike HiPFaR, the baseline method is based on a detour strategy and thereby packets may take unnecessary longer paths to reach destinations.

A. Performance Analysis under Uniform Traffic Profile

In the uniform traffic profile, each processing element (PE) generates data packets and sends them to another PE using a uniform distribution [26]. The mesh size is considered to be 4×4. The average communication latency of HiPFaR and the baseline method are compared with zero- and one-faulty node cases. As observed from the results shown in Fig. 10(a), the performance of both methods is comparable since both methods are not fully adaptive. Although the baseline method is based on a deterministic routing algorithm, it performs slightly better than HiPFaR. The reason is that in the baseline method, packets can switch between the virtual channels in the Y dimension while in HiPFaR the packet adaptivity (between the X and Y dimensions) is limited when the packet gets close to the destination node. But, as the network enlarges both algorithms behave with a similar performance. In a one-faulty case, HiPFaR outperforms the baseline method. This is due to the fact that HiPFaR can route packets through the shortest paths while in the baseline method, packets may take longer paths when facing a faulty node.

B. Performance Analysis under Hotspot Traffic Profile

Under the hotspot traffic pattern, one or more nodes are chosen as hotspots receiving an extra portion of the traffic in addition to the regular uniform traffic. In simulations, given a hotspot percentage of H, a newly generated message is directed to each hotspot node with an additional H percent probability. We simulate the hotspot traffic with a single hotspot node at (2, 2) in 4×4 2D mesh network. The performance of the HiPFaR and the baseline method is measured for fault-free and one-faulty node cases. The performance of each network with $H = 10\%$ is illustrated in Fig. 10(b). As observed from the figure, in the hotspot traffic and in both faulty and non-faulty cases, the performance gain of HiPFaR is larger than the detour-based scheme (baseline).

C. Reliability Evaluation under Uniform Traffic Profile

To evaluate the reliability of HiPFaR, the number of faulty nodes increases from 1 to 6. All faulty nodes are selected using a random function. The results are obtained using 10000 iterations in an 8×8 mesh network when the traffic is uniform random. Reliability is measured based on the number of successful packet arrivals at the destination nodes into the total number of delivered packets. In simulation, we assume that faulty nodes can act as a source and destination node but not as an intermediate node. In other words, they can send and receive packets but packets cannot be passed through them. As shown in Fig. 11, HiPFaR can tolerate up to six faulty nodes by more than 98% reliability.

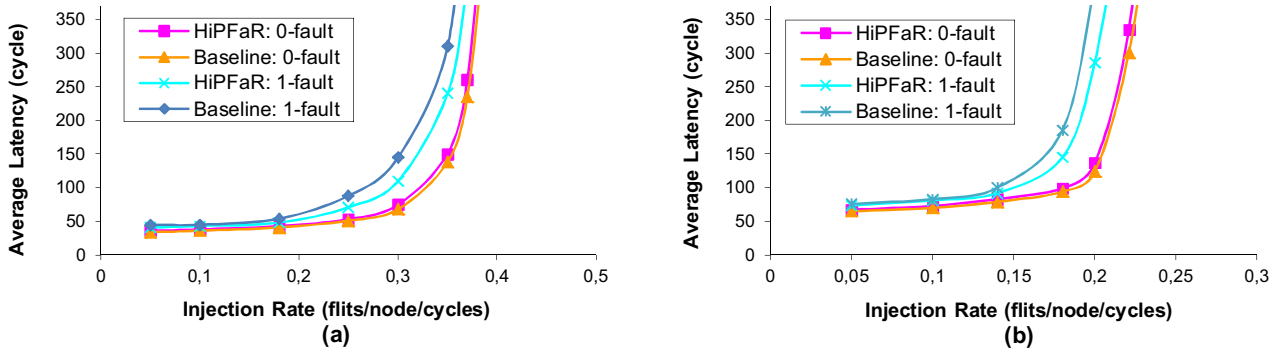

Fig. 10. Performance analysis of HiPFaR and the baseline method in 4×4 mesh network (a) under uniform traffic profile (b) hotspot traffic profile in fault-free and one-faulty cases

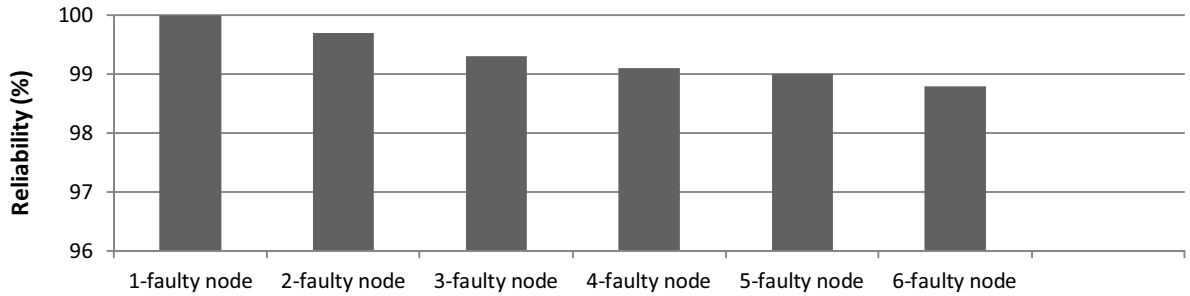

Fig. 11. Reliability evaluation of HiPFaR in 8×8 mesh network under uniform traffic profile

D. Hardware Analysis

To assess the area overhead and power consumption, the whole platform of each method is synthesized by Synopsys Design Compiler. We compared the area overhead and power consumption of HiPFaR with the baseline method. The power consumption of both methods is measured in one-faulty node case. For each scheme we include network interfaces, routers, and communication channels. For synthesizing we use the UMC 90nm technology at the operating frequency of 1GHz 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 a 6×6 2D mesh. The layout area and power consumption of each platform are shown in Table 1. As indicated in the table, HiPFaR has a lower area overhead than the baseline method. It is because of using a more complex router in the baseline method.

Table 1. Details of hardware implementation

Network platforms	Area (mm^2)	Power (W) dynamic & static
HiPFaR	6.794	2.39
baseline	6.903	2.53

VI. CONCLUSION

In this paper, we present a fault-tolerant routing algorithm named HiPFaR. This algorithm is built upon a fully adaptive routing algorithm with the minimum number of virtual channels (i.e. one and two virtual channels along the X and Y dimensions). This idea is general and can be applied to a network with a higher number of virtual channels. HiPFaR uses the shortest paths from the source to the destination node in the faulty network. When the packet is east-, west-, north-, or south-ward, a non-minimal path is necessitated to bypass a fault. It was proven that, the non-minimal paths can be taken in the network without creating any cycle. The analyses under the uniform random and hotspot traffic indicate that HiPFaR maintains the performance of NoC in the presence of faults. Finally, the reliability analyses show that HiPFaR is highly resilient under multiple faulty nodes by more than 98% reliability when there are six faulty nodes in the 8×8 mesh network.

ACKNOWLEDGMENT

The authors wish to acknowledge Nokia, Elisa, and Kaute Foundations for the partial financial support during the course of this research.

REFERENCES

[1] Xu, Jiang, W. Wolf, J. Hankel, S. Charkdhar, "A Methodology for design, modeling and analysis for networks-on-Chip," IEEE International Symposium on Circuits and Systems, pp. 1778-1781, 2005.

[2] M. Daneshtalab et al., "Adaptive Input-output Selection Based On-Chip Router Architecture," Journal of Low Power Electronics *(JOLPE)*, Vol. 8, No. 1, pp. 11-29, 2012.

[3] W. Tsai, D. Zheng, S. Chen, and Y.H. Hu, "A fault-tolerant NoC scheme using bidirectional channel", in Proc. DAC, pp.918-923, 2011.

[4] E. Rijpkema et al., "Trade offs in the design of a router with both guaranteed and best-effort services for networks on chip," in Proc. DATE'03, pp. 350-355, 2003.

[5] M.H Neishaburi et al., "A HW/SW Architecture to Reduce the Effects of Soft-Errors in Real-Time Operating System Services," in Proceedings of IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS), pp.1 - 4, Apr 2007, Poland.

[6] M. H. Neishaburi, Z. Zilic, "An enhanced debug-aware network interface for Network-on-Chip," in proc. of International Symposium on Quality Electronic Design (ISQED), pp. 709-716, 2011.

[7] J. Duato, S. Yalamanchili, L. Ni, "Interconnection networks: an engineering approach", Morgan Kaufmann Publishers, 2003.

[8] M. Ebrahimi et al., "CATRA-Congestion Aware Trapezoid-based Routing Algorithm for On-Chip Networks," in Proceedings of 15th ACM/IEEE Design, Automation, and Test in Europe (DATE), pp. 320-325, 2012.

[9] M. Ebrahimi et al. "HARAQ: Congestion-Aware Learning Model for Highly Adaptive Routing Algorithm in On-Chip Networks," in Proceedings of 6th ACM/IEEE International Symposium on Networks-on-Chip (NOCS), pp. 19-26, 2012.

[10] X. Chang et al. "PARS – An Efficient Congestion-Aware Routing Method for Networks-on-Chip," in Proceedings of 16th IEEE International Symposium on Computer Architecture and Digital Systems (CADS), pp. 166-171, 2012.

[11] M. Dehyadegari et al. "An Adaptive Fuzzy Logic-based Routing Algorithm for Networks-on-Chip," in Proceedings of 13th IEEE/NASA-ESA International Conference on Adaptive Hardware and Systems (AHS), pp. 208-214, 2011.

[12] M. Daneshtalab et al. "Distributing Congestions in NoCs through a Dynamic Routing Algorithm based on Input and Output Selections," in Proceedings of 20th IEEE International Conference on VLSI Design (VLSID), pp. 546-550, 2007.

[13] D. Fick et al. "Vicis: a reliable network for unreliable silicon", in Proc. of Design Automation Conference, pp. 812-816, 2009.

[14] S. Chalasani, R.V. Boppana, "Fault-tolerant wormhole routing algorithms for mesh networks", IEEE Trans on Computers, 44(7):848–64, 1995.

[15] PH. Sui, SD. Wang, "An improved algorithm for fault-tolerant wormhole routing in meshes," IEEE Trans on Computers x;46(9):1040–2, 2011.

[16] S. Park, JH. Youn, B. Bose, "Fault-tolerant wormhole routing algorithms in meshes in the presence of concave faults", in Proc. of International Parallel and Distributed Processing Symposium (IPDPS), p. 633–8, 2000.

[17] Z. Zhang, A. Greiner and S. Taktak, "A reconfigurable routing algorithm for a fault-tolerant 2D-mesh Network-on-Chip", in Proc. DAC, pp. 441-446, 2008.

[18] M. Koibuchi, H. Matsutani, H. Amano, and T.M. Pinkston, "A Lightweight Fault-Tolerant Mechanism for Network-on-Chip", in Proc. NOCS, pp.13-22, 2008.

[19] J. Wu, "A Fault-Tolerant and Deadlock-Free Routing Protocol in 2D Meshes Based on Odd-Even Turn Model", in Proc. IEEE transaction on computers, v. 52, pp.1154-1169 ,2003.

[20] M. Ebrahimi et al., "MAFA: Adaptive Fault-Tolerant Routing Algorithm for Networks-on-Chip," in Proceedings of 15th IEEE Euromicro Conference On Digital System Design (DSD), pp. 201-206, 2012.

[21] M. Ebrahimi, M. Daneshtalab, J. Plosila, F. Mehdipour, "MD: Minimal path-based Fault-Tolerant Routing in On-Chip Networks," in Proceedings of 18th Asia and South Pacific Design Automation Conference (ASP-DAC), 2013.

[22] D. Fick, A. DeOrio, G. Chen, V. Bertacco, D. Sylvester and D. Blaauw, "A highly resilient routing algorithm for fault-tolerant NoCs", in Proc. DATE, pp. 21-26, 2009.

[23] C. Glass and L. Ni, "Maximally Fully Adaptive Routing in 2D Meshes," In Proc. of Parallel Processing, pp.101-104, 1992.

[24] M. Daneshtalab et al. "A Low-Latency and Memory-Efficient On-Chip Network," in Proceedings of 4th ACM/IEEE International Symposium on Networks-on-Chip (NOCS), pp. 99-106, 2010.

[25] M. Ebrahimi et al. "HAMUM – A Novel Routing Protocol for Unicast and Multicast Traffic in MPSoCs," in Proceedings of 18th IEEE Euromicro Conference on Parallel, Distributed and Network-Based Computing (PDP), pp. 525-532, 2010.

[26] C.J. Glass et al., "The Turn Model for Adaptive Routing", in Proc. 19th Int'l Symp. Computer Architecture, pp. 278-287, 1992.