

## Resilient Routing Implementation in 2D Mesh NoC

Rimpy Bishnoi · Vijay Laxmi · Manoj Singh Gaur ·  
Mark Zwolinski

Received: date / Accepted: date

**Abstract** With the rapid shrinking of technology and growing integration capacity, the probability of failures in Networks-on-Chip (NoCs) increases and thus, fault tolerance is essential. Moreover, the unpredictable locations of these failures may influence the regularity of the underlying topology, and a regular 2D mesh is likely to become irregular. Thus, for these failure-prone networks, a viable routing framework should comprise a topology-agnostic routing algorithm along with a cost-effective, scalable routing mechanism able to handle failures, irrespective of any particular failure patterns. Existing routing techniques designed to route irregular topologies efficiently lack flexibility (logic-based), scalability (table-based) or relaxed switch design (uLBDR-based). Designing an efficient routing implementation technique to address irregular topologies remains a pressing research problem. To address this, we present a fault resilient routing mechanism for irregular 2D meshes resulting from failures. To handle irregularities, it avoids using routing tables and employs a few fixed configuration bits per switch resulting in a scalable approach. Experiments demonstrate that the proposed approach is guaranteed to tolerate all locations of single and double-link failures and most multiple failures. Also, unlike uLBDR it is not restricted to any particular switching technique and does not replicate any extra messages. Along with fault tolerance, the proposed mechanism can achieve better network performance in fault-free cases. The proposed technique achieves graceful performance degradation during failure. Compared to uLBDR, our method has 14% less area requirements and 16% less overall power consumption.

**Keywords** NoC · Routing implementation · Fault tolerance · Segment routing

---

R. Bishnoi  
Department of Computer Science and Engineering  
MNIT Jaipur, India  
E-mail: rimpybishnoi@gmail.com

V. Laxmi  
Department of Computer Science and Engineering  
MNIT Jaipur, India  
E-mail: vlgaaur@gmail.com

M. S. Gaur  
Department of Computer Science and Engineering  
MNIT Jaipur, India  
E-mail: gaurms@gmail.com

Mark Zwolinski  
University of Southampton, Southampton, United Kingdom  
E-mail: mz@ecs.soton.ac.uk

## 1 Introduction

Network-on-Chip (NoC) is a scalable and modular communication paradigm proposed to overcome the limitations of traditional bus-based interconnects [1]. Regular network structures like 2D meshes are usually preferred by NoC designers owing to their simplicity and perfect physical layout on a 2D surface. As scaling of CMOS technology approaches nanometre scales, the reliability of on-chip interconnects becomes a design concern, as any failure may cause the entire system to fail in non-fault tolerant designs [2, 3].

In traditional network literature, the most popular fault classes are transient faults and permanent faults [3]. Transient faults include temporary failures (bits-errors in a physical channel) that often occur at random and unpredictable times. Faults occurring due to neutrons and alpha particles are classed as transient faults. Other faults are permanent, including failures that are not temporary and that result in irreversible physical changes. These are caused by poor design, including incorrect specifications, manufacturing defects, component wear-outs, random device effects, broken wires, time-dependent dielectric breakdowns, electromagnetic interference (EMI), etc. [4].

Although, permanent faults are not as common and frequent as transient faults, our focus in this paper is restricted only to permanent failures. The reason is that transient faults can be detected and corrected locally by methods such as cyclic redundancy checking or forward error correction [4]. Moreover, if it occurs frequently, a transient fault can be modelled as a time-limited permanent fault and preventive measures taken for permanent faults are also applicable to this fault class. On the other hand, the effects of permanent faults are irreversible, and it is not always possible to repair or replace the components on a chip [3].

Furthermore, due to decrease of inter-wire spacing in modern on-chip interconnects, short switch-to-switch links in regular structures are more prone to noise sources such as crosstalk, EMI, radiation, process variations and so on. This fact makes links more susceptible to failures [4]. More specifically, our interest is restricted only to the permanent failures of underlying physical links and switches of a 2D mesh NoC. Failures may also be present in other components of a system, however, such as in computational components (core), storage components (cache slice), power supplies, clocks, etc. The failure of computational/storage components may result only in degraded system performance. A failure in any of the NoC components (link or switch) will be more harmful as the NoC provides the communication backbone to connect multiple components on the chip. Indeed, such a failure may become a single point of failure, potentially causing the entire system to fail.

The unpredictable locations of these failures may also harm the regularity of the underlying topology. More specifically, a regular 2D mesh is more likely to become irregular due to these failures. Thus, a NoC must continue its operation even if the network becomes irregular.

In the event of failure, it is possible that the current routing function is unable to offer a path between each pair of nodes. Hence, it needs to be replaced with one offering full connectivity. To ensure this requirement, it is necessary that the underlying routing implementation framework should provide the flexibility to reconfigure the old routing function with a new routing function. This expectation demands a flexible routing framework for NoCs, which can efficiently support any irregularity generated in the initial regular structures.

A finite-state machine (FSM) based implementation [5], is very efficient in terms of both area and latency but is topology- and routing-dependent. Such methods may not work in the event of any failure-induced irregularity. Implementations based on forwarding tables (source, distributed) [6–8] can be used to support any irregular topology. Though tables provide the flexibility to work with any irregular topology, they do not scale well in terms of area, power, and performance.

Recently, uLBDR (Universal Logic Based Distributed Routing) [9] has been proposed as a scalable and efficient routing implementation method to support irregular 2D mesh topologies. It combines the scalability of traditional FSM based implementations and the flexibility of table-based implementations.

Despite its scalability and flexibility, uLBDR adds some limitations to the switch design that restrict its applicability to VCT (Virtual Cut-Through) switching only and it requires a particular arbiter. In addition to this, uLBDR also increases the network traffic by replicating messages [9].

## 1.1 Motivation and Objectives

Most routing implementation techniques that have been proposed to efficiently capture the routing function for irregular topologies are either table based [6–8] or require a complex constrained switch design [9]. This fact motivates us to design an efficient routing implementation technique to address irregular topologies.

The main objective of this paper is to propose a fault-tolerant routing implementation that can handle the irregularities resulting from failures in regular 2D meshes. The aim is to provide the routing framework with the flexibility to reconfigure the routing function when the topology becomes irregular. In addition to this, the proposed implementation should achieve high performance (low latency, high throughput) while keeping the cost and complexity low.

## 1.2 Contributions

To achieve the aforementioned objective, we propose a routing implementation technique. Our proposal is capable of handling any irregularity induced in a 2D mesh because of any number of 1-link and 2-link and most multiple link failures. The novelty of our proposal lies in the fact that it neither utilizes any routing tables nor imposes any restrictions on switch design. Additionally, it is deadlock and livelock free. Having these properties, the proposed method does not require any additional hardware (virtual channels). While developing an efficient routing implementation, the proposed mechanism is designed to achieve the following:

1. **Improved fault coverage:** This is a measure of the reliability of the proposed approach. It is defined as the percentage of irregular topologies (generated due to link failures) supported by a particular routing implementation. The proposed implementation guarantees full coverage in the case of any number of 1-link and 2-link failures provided there exists a deadlock-free path from source to destination. We have also extended the scope of the proposed mechanism to handle multiple failures.
2. **Performance:** In the case of a regular 2D mesh structure (without any failures), this approach maintains low latency and achieves high throughput similar to the baseline technique. In the cases of irregular structures being generated due to failures in an initial 2D mesh, the proposed approach gracefully degrades the network's performance.
3. **Minimized area and power overhead:** For applicability in the NoC domain, a low value for both area and power is necessary while keeping the design scalable. The proposed method requires respectively 14% and 16% less area and power than other state-of-the-art logic-based solutions proposed for irregular topologies and it keeps the design scalable.

The rest of the paper is organized as follows: In Section 2 we discuss the related work. In Section 3, we describe our proposed mechanism. We present experimental results analysis in Section 4. Finally, concluding remarks are presented in Section 5.

## 2 Related Work

Considerable research has been carried out on resilient NoCs. In this section, we discuss previous work targeting permanent failures (either links or switches). We classify solutions based on their routing implementation mechanism, indicating whether they support fault tolerance or not.

1. **FSM based implementations:** An FSM based implementation [5] of a routing algorithm is very efficient in terms of both performance and area but is topology sensitive. Any failure in the network might convert a regular topology into an irregular one. There is a large body of work on fault tolerance based on FSM implementations providing only partial support for irregular topologies. They are not able to handle all possible sets of irregular topologies derived from a specific set of failures. Some of them utilize virtual channels to implement fault-tolerant routing algorithms. The adaptive routing algorithm proposed by Linder and Harden [10] supports single node failure by doubling the number of virtual channels along one dimension, resulting in underutilized resources. The reliable router [11], proposed by Dally et al for 2D meshes, can support irregular topologies derived from a single link or node failure. To achieve this, it also utilizes a large number (five for each physical link) of virtual channels. Similarly, several other prior works [12,13] utilize additional virtual channels to provide fault tolerance and result in improved coverage support. In [14], Glass and Ni showed how to modify turn model-based routing algorithms to provide  $(n - 1)$  fault tolerance for  $n$ -dimensional meshes without using any virtual channels.
2. **Table based implementations:** Implementations based on forwarding tables (either at source or distributed) are not sensitive to topology change and offer the flexibility to support any irregular topology derived from any set of link failures. Schonwald et al [15] proposed a table-based Force-Directed Wormhole Routing (FDWR) which is based on hop distance to the destination from the current switch. Though it supports all irregular topologies as tables are deployed, it results in a large packet processing time at routers. Large table size is also an issue with FDWR. Feng et al [8] proposed a Fault Tolerant Deflection Router (FTDR) able to handle permanent and transient faults. They also proposed an improvement over the basic FTDR algorithm, named FTDR-H, which tries to reduce the table size by dividing the network into regions. However, the problem of increasing the number of tables with network size remains the same. A few other works on fault tolerance [6,7] also employ tables either at the source or a router. A number of resilient routing algorithms based on network reconfiguration have also been proposed such as Immunet [16], Vicis [17] and ARIADNE [18]. Immunet [16] routes packets towards their destination using a fully adaptive routing algorithm. A new ring is used for network reconfiguration and deadlock freedom that deterministically routes packets to their destinations. A major issue with this approach is that it requires three routing tables per router, which drastically increases the area overhead. Based on the odd-even turn model, Fick et al [17] proposed a low overhead routing algorithm called Vicis to handle unlimited failures. However, it is not clear that the algorithm remains deadlock-free in the face of many failures because reconfiguration re-enables turns prohibited by the routing algorithm. ARIADNE [18] is a similar approach based on the reconfiguration of routing tables. It utilizes the up\*/down\* routing algorithm and reconfigures it when faults occur. Though ARIADNE provides an area improvement over Immunet, its underlying routing algorithm is not optimized for regular networks. Thus, solutions based on forwarding tables are flexible for irregular topologies but suffer from the scalability and high cost associated with tables [9].
3. **LBDR based implementation:** To keep the implementation flexible, as forwarding tables, while offering scalability as in FSM based implementations, Rodrigo et al [19] proposed Logic Based Distributed Routing (LBDR). It is a cost effective routing implementation mechanism that is based on a few configuration bits (routing and connectivity) and fixed logic per switch. LBDR offers fault tolerance by allowing these bits to be configured according to the current routing function and topology in the event of any topology change. It offers full support for all irregular topologies derived from 2D mesh as long as there exists a minimal path between each source and destination pair. In the case where no minimal path exists between a pair of nodes, LBDR fails to handle that particular irregular topology.

To support this, Rodrigo et al [9] presented Universal Logic Based Distributed Routing (uLBDR). uLBDR ensures complete reachability by packet replication at two output ports which leads to a few design constraints on the network switch. The first is that packet replication increases the chance of having deadlock when used with wormhole switching. Thus, the Virtual-Cut-Through (VCT) approach is used. Second, a traditional arbiter needs to be modified to allow a single message to complete at more than one output port simultaneously. In addition to this, packet replication potentially leads to an increase in network traffic.

In this work, we propose a routing implementation that does not use any routing tables to capture the routing function and is thus scalable. Like LBDR, it captures the routing function using few configuration bits. Unlike uLBDR, it does not impose any additional switch design constraints. Also, its implementation requires smaller area than uLBDR while maintaining the same performance and coverage.

### 3 Proposed Work

In this section, we first define the terminologies associated with 2D mesh NoCs. Then the proposed approach is explained in detail through illustrative examples. Finally, we present the deadlock freedom and connectivity proof mandatory for any efficient implementation.

#### 3.1 Definitions and Concepts

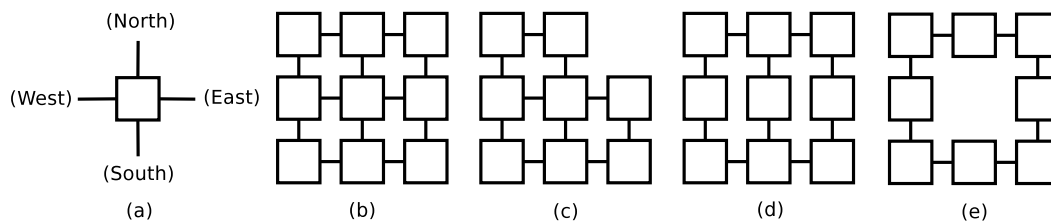
**Definition 1:** An interconnection network, say an NoC, is a strongly connected graph,  $G(N, L)$ , where  $N$  represents the set of processing nodes, and  $L$  represents the set of bidirectional communication links through which processing nodes communicate.

**Definition 2:** A regular 2-dimensional (2D) mesh NoC is formally defined as an interconnection network that has  $m \times n$  processing nodes ( $m$  is no. of rows and  $n$  is no. of columns). Any node in the mesh is identified by coordinates  $(r, c)$ ;  $0 \leq r \leq m - 1$ ,  $0 \leq c \leq n - 1$ ,  $r$  and  $c$  are called the  $X$  coordinate and  $Y$  coordinate, respectively.

**Definition 3:** An irregular 2D mesh NoC  $G'(N', L')$  derived after fault induction in a regular 2D mesh NoC  $G(N, L)$ , is a subgraph  $G'$  of  $G$ , where  $N' \subseteq N$  and  $L' \subseteq L$ .

**Definition 4:** Topology Coverage is defined as the percentage of irregular 2D mesh topologies supported out of the overall number of irregular 2D mesh topologies that can be generated from a regular 2D mesh because of failures. A topology is said to be supported if there exists at least a path between each pair of end-nodes.

**Definition 5:** Fault tolerance is the ability of an interconnection network to continue operations (network service) even in the presence of one or more component failures.



**Fig. 1** (a) Port directions in 2D Mesh (b-e) Examples of regular (b) and irregular (c, d, e) 2D mesh topologies

For on-chip interconnects, a 2D mesh is considered over other topology choices due to its simplified design and perfect physical layout on a 2D surface. In a regular 2D mesh, each switch is connected to its

neighbouring switches by four bidirectional ports/channels labelled as  $N$  (North),  $E$  (East),  $S$  (South), and  $W$  (West), as shown in Figure 1(a). Any failure (either switch or link) in the initial regular 2D mesh makes the topology irregular.

Figure 1 shows examples of irregular topologies generated from a regular 2D mesh (the three right-most figures). Solutions proposed for regular 2D meshes may not work once the topology becomes irregular. To handle this, we propose a fault-resilient routing implementation technique that can handle regular as well as all irregular topologies generated due to single, double and most of the multiple link failures in a 2D mesh. We consider irregular topologies derived from permanent failures in regular 2D meshes.

We assume the fault detection, notification and updating of corresponding configuration bits at switches occur prior to the start of a normal operation. The centralized infrastructure proposed in [20] can be used for fault detection, notification, and updating of bits. Basically, it employs a global controller with full visibility of the network. In the event of any changes in the network (either topology or routing), the global controller is in charge of computing a new set of configuration bits. To carry the vital information from NoC switches to the controller (testing, diagnosis data) and vice versa (updated configuration bits), a cost-effective dual network is employed [20].

### 3.2 Methodology

The routing function of any algorithm determines a set of effective paths that a message needs to follow from its source to destination. More specifically, the choice of routing algorithm greatly affects the network performance. For regular topologies like 2D meshes, topology-dependent routing algorithms, like dimension order routing algorithms, result in reduced latency, power, and area requirements. But as these algorithms are sensitive to topology changes, they may not work if the topology becomes irregular due to a failure. Hence, topology-agnostic routing algorithms, which are not sensitive to any topology changes are used for fault-prone networks [21, 22].

In this work, we have used segment-based routing (SR), a topology-agnostic routing algorithm. Unlike other topology-agnostic routing algorithms such as up\*/down\* [23], SR routing exploits the regularity of 2D meshes and also performs well under regular as well as irregular topology scenarios [21, 22]. However, deadlock-freedom is an essential characteristic of any routing algorithm and to ensure this property each routing algorithm restricts certain turns. These restricted turns, also called routing restrictions, indicate the pair of adjacent channels that cannot be taken by a message, one after the other. Figure 2 shows routing restrictions defined by  $XY$  and SR routing algorithms on a  $3 \times 3$  mesh. Each arrow represents a routing restriction in a particular direction. As can be seen in Figure 2(b), SR routing applies bidirectional routing restrictions that restrict the pairs of channels that cannot be used in both directions.

As mentioned earlier in section 1, our proposed implementation does not require any routing tables to implement the routing algorithm. Similar to LBDR [19], it uses a few fixed configuration bits per switch for this purpose. Unlike tables, the number of bits per switch does not increase with network size; hence the approach is scalable. To capture these routing restrictions defined by SR routing and the connection pattern of a topology, the proposed approach requires three categories of configuration bits for its functioning, as shown in Figure 3. These sets are discussed below

1. Routing bits ( $R_{pq}^{sId}$ ): Routing bits are used to capture the routing restrictions of a routing algorithm located at a switch  $sId$ . As in distributed routing, to advance a particular packet towards some output, every switch needs to know whether the packet can be allowed to take the required turn at the next switch or not. This information is needed to select the correct port direction as the turn might be restricted for deadlock-freedom purpose. For example, to forward a packet in the North direction, the current switch needs to know whether the packet can turn towards East at the next switch or not. More specifically, bit  $R_{pq}^{sId}$  at any switch  $sId$ , indicates if a packet outbound in direction  $p$  can be routed to a direction  $q$  at the next switch or not. Three bits per switch output port are used. The routing bits

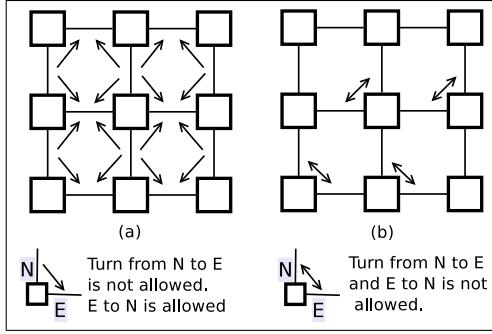


Fig. 2 Routing Restrictions of (a) XY (b) and SR routing

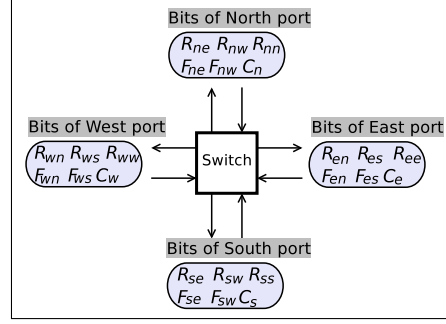


Fig. 3 Configuration bits of a switch (Routing, Connectivity, and Faulty)

for a North Port are labelled as  $R_{ne}$ ,  $R_{nw}$ , and  $R_{nn}$ . A value of 1 means that routing is permitted. Permitted routing implies that a packet routed in the  $N$  direction can continue  $N$  or turn to  $E$  or  $W$  at the next switch. Similarly, routing bits for the East port are labelled as  $R_{en}$ ,  $R_{es}$ ,  $R_{ee}$ , for South as  $R_{se}$ ,  $R_{sw}$ ,  $R_{ss}$  and for West as  $R_{wn}$ ,  $R_{ws}$ ,  $R_{ww}$ . To capture the available routing permissions at a switch, a total of twelve  $R_{pq}$  bits (3 bits per output port) are needed. For any node  $\eta$ , we can define

$$\begin{aligned} \eta_p &= \text{Neighbour of } \eta \text{ in direction } p, \\ \eta_q &= \text{Neighbour of } \eta_p \text{ in direction } q \\ \rho &= \text{path } \eta \rightarrow \eta_p \rightarrow \eta_q \\ R_{pq} &= \begin{cases} 1 & \Rightarrow \rho \text{ permitted} \\ 0 & \Rightarrow \rho \text{ not permitted} \end{cases} \end{aligned}$$

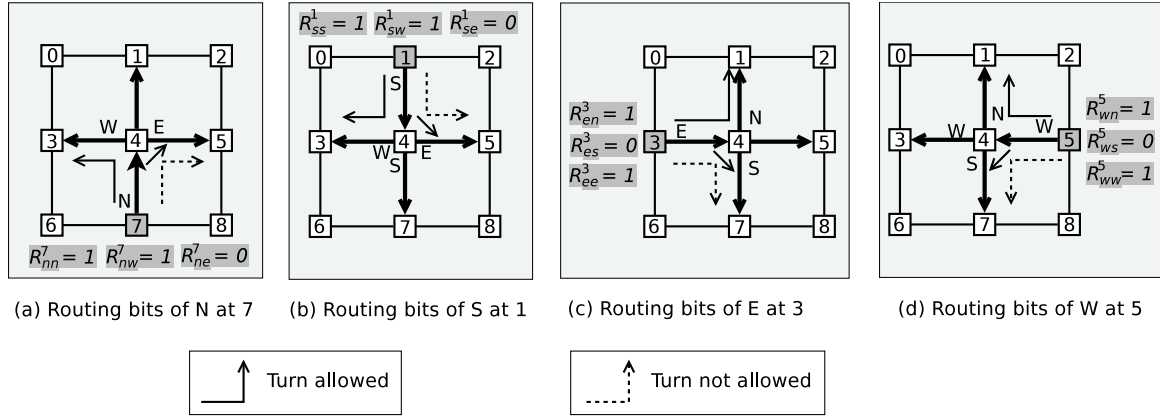


Fig. 4 Routing bits: Illustrative example

Figure 4 shows the snapshot of routing bits at each switch, where each sub-figure demonstrates routing bits of only one port. It can be observed from the figure that the routing bit corresponding to a restricted turn is set to 0; otherwise it is set to 1.

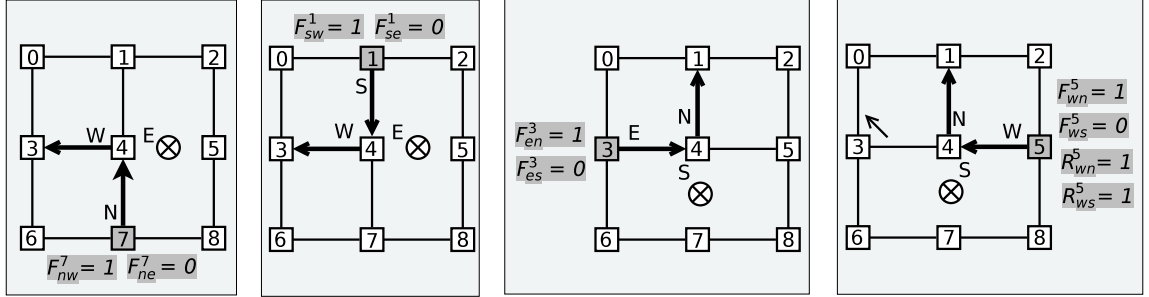
2. Connectivity bits ( $C_p^{sId}$ ): As described above, the rules of the routing algorithm are captured using the routing bits. Similarly, to capture the connection pattern of a particular instance of a topology (either a regular or an irregular 2D mesh), connectivity bits are used. As there are at most four directions

through which a switch,  $sId$  can be connected to its neighbours, we need 4 bits  $C_N, C_E, C_W, C_S$ , with a single bit per output port. These bits indicate the connectivity of a particular switch to its neighbours in *North, East, West, and South* respectively. A value of 1 indicates that the switch is connected to its neighbouring switch in that particular direction. A value of 0 means that connectivity in that particular direction is not present for any reason. For any node  $\eta$ , we can define

$$C_p = \begin{cases} 1 & \text{if } \eta \rightarrow \eta_p \text{ exists} \\ 0 & \text{otherwise} \end{cases}$$

3. Faulty bits ( $F_{pq}^{sId}$ ): Similar to the routing bits, faulty bits indicate the status of the links at a neighbouring switch. In a 2D mesh, at any switch, for all destinations present in any of its quadrant (NE, NW, SE, SW), at most two port directions lead to minimal paths. If any one of the minimal path becomes faulty due to the fault present at neighbouring switch, another can be taken to avoid unnecessary non-minimal paths. This allows a switch to decide which neighbour to select to avoid faults at neighbouring switches. To capture faulty turns, a switch needs two bits per switch output port, a total of eight bits per switch. Faulty bits for a North port are labelled as  $F_{ne}, F_{nw}$ . A value of 1 represents that turn at the next switch is allowed whereas a value of 0 represents a turn that is faulty and, hence not allowed. For example,  $F_{ne} = 0$  means the turn from the North of the current switch to the East of the next switch is faulty. Similarly, faulty bits for the East port are labelled as  $F_{en}, F_{es}$ , for the South as  $F_{se}, F_{sw}$  and for the West as  $F_{wn}, F_{ws}$ . We have not used faulty bits for the cases where the destination is present in same row or column of the source switch ( $F_{nn}, F_{ee}, F_{ss}, F_{ww}$ ). This is because, when both source and destination switches are present in same row or column, a single minimal path exists. Any fault in this path will certainly lead to a non-minimal path. For any node  $\eta$  we can define

$$F_{pq} = \begin{cases} 0 & \text{if } \eta_p \nrightarrow \eta_q \\ 1 & \text{if } \eta_p \rightarrow \eta_q \end{cases}$$



(a) Faulty bits of N at 7 (b) Faulty bits of S at 1 (c) Faulty bits of E at 3 (d) Faulty bits of W at 5

**Fig. 5** Faulty bits: Illustrative example

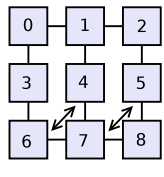
Figure 5 shows a snapshot of faulty bits at each switch, where each sub-figure demonstrates faulty bits of one port only. We can observe from the figure that the faulty bit corresponding to a faulty turn at the next switch is set to zero. In the case of a healthy turn, the bit is set to one. For example, in Figure 5 (a),  $F_{ne}^7$  is set to zero because  $C_e^4$  is zero. On the other hand,  $F_{nw}^7$  is set to one as  $C_w^4$  is also one. Figure 5 (d) shows the purpose of faulty bits.



As shown in Figure 5 (d), both  $R_{wn}^5$  and  $F_{wn}^5$  are set to one due to the absence of any restriction or fault at the North port of the next switch, 4, i.e.  $C_n^4$  is equal to one. However,  $R_{ws}^5$  is set to one, but  $F_{ws}^5$  is set to zero because  $C_s^4$  is faulty. Hence, the turn  $W \leftrightarrow S$  at switch 5 can be made unavailable for switch 7, whereas it can be made available for distant switches, such as 6 in this example. Hence, faulty bits are used to distinguish between a turn that might be available for some distant switches but might not be available for a nearby switch due to a fault at the neighbouring switch.

Table 1 shows all the configuration bits of the proposed method calculated for the SR routing algorithm applied to the  $3 \times 3$  irregular mesh of Figure 6. As can be seen, the  $C_n$  bits of switches 0, 1, and 2 are all set to zero as no connectivity exists in the North direction of these switches. Bidirectional arrows in Figure 6 represent restricted routing turns of SR to prevent cycles. As explained above, these routing rules are encoded in routing bits  $R_{pq}$  by setting them to zero or one for prohibited and allowed turns respectively.

For example,  $R_{sw}^4$  and  $R_{en}^6$  are set to zero. This ensures that the packets Southbound from 4 cannot turn West at 7. Similarly, packets Eastbound from 6 cannot turn North at 7. Further, to indicate faults present at 1-hop neighbors, the faulty bits are reset. At switch 0, faulty bit  $F_{se}$  is set to zero to indicate that link  $3 \rightarrow 4$  is broken. Though  $F_{se}^0$  is reset to zero,  $R_{se}^0$  is still set to 1 to allow paths to switches 7, 8 through switch 3 in the presence of the  $3 \rightarrow 4$  broken link. In this way a faulty bit is used to distinguish between a turn that might be available for some distant switches (switch 7, 8) but might not be available for a nearby switch (4) due to the presence of a fault at a neighbouring switch (3).



$sId$	$C_n$	$C_e$	$C_w$	$C_s$	$R_{nn}$	$R_{ne}$	$R_{nw}$	$R_{ee}$	$R_{en}$	$R_{es}$	$R_{ww}$	$R_{wn}$	$R_{ws}$	$R_{ss}$	$R_{se}$	$R_{sw}$	$F_{ne}$	$F_{nw}$	$F_{en}$	$F_{es}$	$F_{wn}$	$F_{ws}$	$F_{se}$	$F_{sw}$
0	0	1	0	1	0	0	0	1	0	1	0	0	0	1	1	0	0	0	0	1	0	0	0	0
1	0	1	1	1	0	0	0	0	0	1	0	0	1	1	1	0	0	0	0	1	0	1	0	0
2	0	0	1	1	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	1	0	0
3	1	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	1	0
4	1	0	0	1	0	1	1	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	1	0
5	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
6	1	1	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	1	1	1	0	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0
8	1	0	1	0	1	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	1	0	0	0

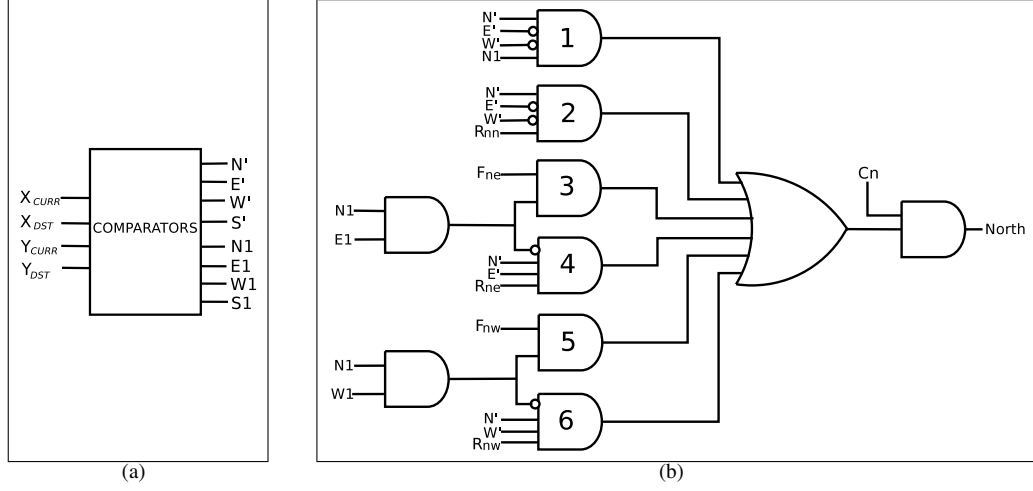
**Fig. 6**  $3 \times 3$  Irregular mesh **Table 1** Configuration bits for SR algorithm for a  $3 \times 3$  mesh topology

Furthermore, this distinguishes our approach from the uLBDR approach [9]. For example, on the failure of the East port of a switch 3, uLBDR sets  $R_{se}^0$  to zero. This results in path  $0 \rightarrow 3 \rightarrow 6 \rightarrow 7 \dots$  being excluded from future considerations. As stated above, handling faults at nearby nodes may render routes to distant nodes unavailable in uLBDR. We incorporate faulty bits  $F_{pq}^{sId}$  for each switch ( $sId$ ) to avoid this. These bits indicate faults at 1-hop neighbours and restrict only routes to such neighbours. For others, the route can be used as long as there are no routing restrictions.

Furthermore, it is important to note that if a turn towards some output direction is not allowed at any of its neighbouring switches present in the same row or column, both routing and faulty bits are reset. For example, unlike  $R_{se}^0$ ,  $R_{sw}^2$  is set zero because similar turns are also prohibited at the next switches present in the same column. In this case both  $R_{sw}^2$  and  $F_{sw}^2$  are set to zero.

### 3.3 Proposed Routing Logic

The routing algorithm defines the rules of routing a packet, and the routing implementation mechanism applies these rules to generate a valid output port direction for an incoming packet. Our proposed imple-



**Fig. 7** Routing Logic (a) First level: Comparator (b) Second level: Routing computation for *North* output port

mentation uses the SR routing algorithm. It captures the routing algorithm and connection pattern of a topology as a set of configuration bits per switch as described in the previous section.

Figure 7 shows the proposed two-level routing logic. Algorithm 1 shows the corresponding algorithm for generating possible output port directions for an incoming packet at current switch  $c$ . The first level is a comparator, as shown in Figure 7 (a) that outputs the direction (directions) of a destination switch relative to the position of the current switch. As shown in Algorithm 1, first coordinates of the current and destination switch are compared. Depending on the location of the destination switch, one or at most two signals from  $\{N', E', W', S'\}$  will be set for any packet. For example, if the current switch of a packet is in the same row as the destination switch,  $\{E', W'\}$  will be activated. Similarly, if the destination switch is in  $NE$  quadrant of the current switch,  $\{N', E'\}$  will be activated. In addition, the respective signal from  $\{N1, E1, W1, S1\}$  is also set if the destination is 1-hop in the given direction. For example, a destination in the  $NE$  quadrant is indicated by  $N' = E' = 1, W' = S' = 0$ . If this destination is 1-hop away in the  $NE$  quadrant, then signals  $N1 = E1 = 1, W1 = S1 = 0$  will be set.

After knowing the destination's quadrant direction, the second level of the logic performs routing and generates output port directions that can be used by a packet for traversing to the next switch. This logic makes use of configuration bits  $\{R_{pq}^c, C_p^c, F_{pq}^c\}$  of the current switch  $c$ . Due to the similarity in the logic of all port directions, we will discuss in detail the routing logic of the North port as shown in Figure 7 (b) and lines 1  $\rightarrow$  5 of Algorithm 1. At current switch  $c$ , the North port is selected for routing if its  $C_n^c$  is one and any one of the following conditions <sup>1</sup> is true:

1. The destination switch is in the same column and only 1-hop away in the North direction from the current switch ( $N' \wedge \overline{E'} \wedge \overline{W'} \wedge N1$ ).
2. The destination switch is in the same column but more than 1-hop away in the North direction from the current switch and routing North is allowed at the next switch, i.e.  $R_{nn}$  of the current switch is set to one ( $N' \wedge \overline{E'} \wedge \overline{W'} \wedge R_{nn}$ ).
3. The destination is in the  $NE$  quadrant from the current switch and 1-hop away in the North and East directions, and  $F_{ne} = 1$ , i.e. there is no Eastward fault at the next switch ( $N1 \wedge E1 \wedge F_{ne}$ ).
4. The destination is in the  $NE$  quadrant from the current switch but more than 1-hop away in the North and East directions, and  $R_{ne} = 1$  at the current switch, i.e from the North of the current switch, going East is allowed at the next switch ( $N' \wedge E' \wedge (\overline{N1} \wedge \overline{E1}) \wedge R_{ne}$ ).

<sup>1</sup> Each condition corresponds to one AND gate of Figure 7 (b)

**Algorithm 1** Algorithm for generating port directions**Require:**  $X_c, Y_c$ :  $X$  and  $Y$  coordinates of current switch  $c$  $X_d, Y_d$ :  $X$  and  $Y$  coordinates of destination switch  $d$  $R_{pq}^c, C_p^c, F_{pq}^c$ : Routing, connectivity, and faulty bits of current switch  $c$ **Ensure:**  $N', E', S', W'$ : Direction of destination switch $N1, E1, S1, W1$ : One hop away information of destination switch $Port_{c \rightarrow d}$  [4]: Port directions ( $N, E, S, W, or L$ ) at  $c$  for destination  $d$ **Procedure** $L = (X_d == X_c) \text{ AND } (Y_d == Y_c)$  $N' = (Y_d < Y_c), E' = (X_d > X_c), S' = (Y_d > Y_c), W' = (X_d < X_c)$  $N1 = (Y_d == Y_c - 1), S1 = (Y_d == Y_c + 1)$  $E1 = (X_d == X_c + 1), W1 = (X_d == X_c - 1)$ 

```

1: if  $C_n^c$  then
2:   if  $(N' \wedge \overline{E'} \wedge \overline{W'} \wedge N1)$  OR  $(N' \wedge \overline{E'} \wedge \overline{W'} \wedge R_{nn}^c)$  OR  $(N' \wedge E' \wedge (\overline{N1} \wedge \overline{E1}) \wedge R_{ne}^c)$ 
      OR  $(N1 \wedge E1 \wedge F_{ne}^c)$  OR  $(N1 \wedge W1 \wedge F_{nw}^c)$   $(N' \wedge W' \wedge (\overline{N1} \wedge \overline{W1}) \wedge R_{nw}^c)$  then
3:      $Port_{c \rightarrow d}[N] = 1$  {Generate North direction}
4:   end if
5: end if
6: if  $C_e^c$  then
7:   if  $(E' \wedge \overline{N'} \wedge \overline{S'} \wedge E1)$  OR  $(E' \wedge \overline{N'} \wedge \overline{S'} \wedge R_{ee}^c)$  OR  $(E' \wedge N' \wedge (\overline{E1} \wedge \overline{N1}) \wedge R_{en}^c)$ 
      OR  $(E1 \wedge N1 \wedge F_{ne}^c)$  OR  $(E1 \wedge S1 \wedge F_{es}^c)$  OR  $(E' \wedge S' \wedge (\overline{E1} \wedge \overline{S1}) \wedge R_{es}^c)$  then
8:      $Port_{c \rightarrow d}[E] = 1$  {Generate East direction}
9:   end if
10: end if
11: if  $C_s^c$  then
12:   if  $(S' \wedge \overline{E'} \wedge \overline{W'} \wedge S1)$  OR  $(S' \wedge \overline{E'} \wedge \overline{W'} \wedge R_{ss}^c)$  OR  $(S' \wedge E' \wedge (\overline{S1} \wedge \overline{E1}) \wedge R_{se}^c)$ 
      OR  $(S1 \wedge E1 \wedge F_{se}^c)$  OR  $(S1 \wedge W1 \wedge F_{sw}^c)$   $(S' \wedge W' \wedge (\overline{S1} \wedge \overline{W1}) \wedge R_{sw}^c)$  then
13:      $Port_{c \rightarrow d}[S] = 1$  {Generate South direction}
14:   end if
15: end if
16: if  $C_w^c$  then
17:   if  $(W' \wedge \overline{N'} \wedge \overline{S'} \wedge W1)$  OR  $(W' \wedge \overline{N'} \wedge \overline{S'} \wedge R_{ww}^c)$  OR  $(W' \wedge N' \wedge (\overline{W1} \wedge \overline{N1}) \wedge R_{wn}^c)$ 
      OR  $(W1 \wedge N1 \wedge F_{wn}^c)$  OR  $(W1 \wedge S1 \wedge F_{ws}^c)$  OR  $(W' \wedge S' \wedge (\overline{W1} \wedge \overline{S1}) \wedge R_{ws}^c)$  then
18:      $Port_{c \rightarrow d}[W] = 1$  {Generate West direction}
19:   end if
20: end if

```

5. The destination is in the  $NW$  quadrant from the current switch and 1-hop away in the North and West directions, and  $F_{nw} = 1$ , i.e. there is no Westward fault at the next switch ( $N1 \wedge W1 \wedge F_{nw}$ ).
6. The destination is in the  $NW$  quadrant from the current switch but more than 1-hop away in the North and West directions, and  $R_{nw} = 1$  at the current switch, i.e from the North of the current switch, going West is allowed at the next switch ( $N' \wedge W' \wedge (\overline{N1} \wedge \overline{W1}) \wedge R_{nw}$ ).

Similarly, as shown in Algorithm 1 (lines 6  $\rightarrow$  20), the logic conditions for other directions (East, South, and West) can also be determined.

### 3.4 Selection Function

Along with the routing logic, the proposed mechanism also uses a selection function that tries to reduce the probability of selecting a faulty path. When the routing logic described in Algorithm 1 generates more than one routing options, the selection function selects one out of all available options. As the port

directions generated by the routing logic might lead to some faulty paths, the selection function needs to select the one least likely to encounter a fault along the way from the source to the destination switch.

The selection function of the proposed mechanism is based on the principle that when more than one routing option is available, it selects the direction that does not reduce the distance to the destination in that particular direction to zero. For example, routing logic may generate both North and East as routing options to the destination switch present in  $NE$ . If the destination is at the same distance in the North and East directions from the source, *any* of the port directions (either  $N$  or  $E$ ) can be selected. On the other hand, if taking the  $N$  port reduces the distance to the destination in the North direction to zero whereas  $E$  does not,  $E$  will be preferred to  $N$ . This is because after reducing the distance to zero in one direction, any failure present in that direction would lead to unnecessary non-minimal paths. In some cases, even a non-minimal path may also not be available in that direction.

---

**Algorithm 2** Algorithm for selecting final port direction for routing a packet

---

**Require:**  $X_c, Y_c$ :  $X$  and  $Y$  coordinates of current switch  $c$

$X_d, Y_d$ :  $X$  and  $Y$  coordinates of destination switch  $d$

$Port_{c \rightarrow d}$  [4]: Port directions ( $N, E, S, W, or L$ ) at  $c$  for destination  $d$

**Ensure:**  $Port_{c \rightarrow d}$ : Final port direction ( $N, E, S, W, or L$ ) at  $c$  for destination  $d$

**Procedure**

```

1: if ( $X_d > X_c$ ) then
2:    $\Delta_x = (X_d - X_c)$ 
3: else
4:    $\Delta_x = (X_c - X_d)$ 
5: end if
6: if ( $Y_d > Y_c$ ) then
7:    $\Delta_y = (Y_d - Y_c)$ 
8: else
9:    $\Delta_y = (Y_c - Y_d)$ 
10: end if
11: if ( $Port_{c \rightarrow d} [N] = 1$ ) AND ( $Port_{c \rightarrow d} [E] = 1$ ) then
12:   if ( $\Delta_x == 1$ ) AND ( $\Delta_y > 1$ ) then
13:      $Port_{c \rightarrow d} = N$  {Return North direction}
14:   else if ( $\Delta_y == 1$ ) AND ( $\Delta_x > 1$ ) then
15:      $Port_{c \rightarrow d} = E$  {Return East direction}
16:   else
17:      $Port_{c \rightarrow d} = ANY$  ( $N$  OR  $E$ ) {Return either North or East direction}
18:   end if
19: end if
20: if ( $Port_{c \rightarrow d} [S] = 1$ ) AND ( $Port_{c \rightarrow d} [W] = 1$ ) then
21:   if ( $\Delta_x == 1$ ) AND ( $\Delta_y > 1$ ) then
22:      $Port_{c \rightarrow d} = S$  {Return South direction}
23:   else if ( $\Delta_y == 1$ ) AND ( $\Delta_x > 1$ ) then
24:      $Port_{c \rightarrow d} = W$  {Return West direction}
25:   else
26:      $Port_{c \rightarrow d} = ANY$  ( $S$  OR  $W$ ) {Return either South or West direction}
27:   end if
28: end if

```

---

Algorithm 2 shows the steps for selecting the final port direction for routing a packet at switch  $c$ , when the Algorithm 1 generates more than one routing option. Algorithm 2 first computes the difference in the  $X$  ( $\Delta_x$ ) and  $Y$  ( $\Delta_y$ ) dimensions of the current and destination switches. After difference computation, the algorithm checks both quadrant directions generated by Algorithm 1. Algorithm 2 compares

the difference in each of the directions from the destination. If the difference in one direction is one and in another direction is more than one, the direction having the greater distance will be chosen. In the case where the difference in both directions is more than one, or equal to one, any of the directions can be chosen randomly. We will explain Algorithm 2 for one quadrant as follows:

- **Line 11:** Algorithm 2 generates both  $N$  and  $E$  candidate directions for any destination in  $NE$ .
- **Lines 12-13:** If the difference is one to the East and more than one in the North direction, North will be preferred over East.
- **Lines 14-15:** If the difference is one to the North and more than one in the East direction, East will be preferred over North.
- **Lines 16-18** However, if the difference is one in both North and East directions, either East or North can be selected randomly.

### 3.5 Fault Tolerance

As described in Section 3.1, fault tolerance is the capability of an interconnection network to continue routing operations even in the presence of one or more component failures. The proposed implementation incorporates fault tolerance by providing the flexibility to reconfigure the configuration bits according to the underlying routing algorithm and topology in the event of some topology changes.<sup>2</sup> Also, using the routing logic (Algorithm 1) and selection function (Algorithm 2), our approach successfully handles all the topologies generated by single and double-link failures. We then adapted the mechanism to handle multiple link failures.

#### 3.5.1 Single Link Failure

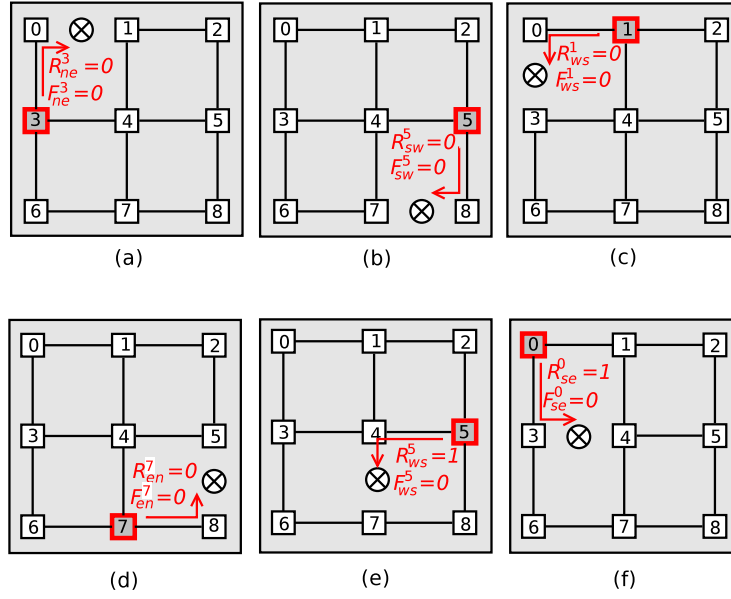
In this section, we will discuss how our proposed implementation handles irregular 2D mesh topologies derived from a single link failure in a regular 2D mesh topology. Assuming different locations for each failure, a number of irregular topologies can be derived. For example, from a  $4 \times 4$  mesh having 24 links, a total of 24 irregular topologies with a single link failure can be generated. Similarly, from a  $3 \times 3$  mesh having 12 links, a total of 12 irregular topologies with a single link failure can be generated. As shown in Figure 8, a single link failure in a 2D mesh topology of any size (in this example a  $3 \times 3$  mesh) can be present either at the boundary (upper, lower, left, and right side) or at the interior of the mesh.

Figure 8 (a) shows the example of a single link present at the upper side boundary of the  $3 \times 3$  mesh. This link failure is located East of switch 0 and West of switch 1. To capture the current topology, the connectivity bits of particular switch output ports connecting the link are set to zero ( $C_e^0, C_w^1$ ). To avoid the failure, switch 3 resets  $F_{ne}^3$  to zero as the turn from the North port of switch 3 to the East port of switch 0 is a faulty turn. The routing bit  $R_{ne}^3$  is also set to zero in this case because the failed link is present at the boundary of the mesh, and the routing turn coded in  $R_{ne}^3$  does not lead to any other path.

Like switch 3, switch 4 updates its faulty and routing bits ( $F_{nw}^4, R_{nw}^4$  equal to zero) accordingly. Setting the routing and faulty bits corresponding to a failure to zero will allow the routing logic to discard the particular port direction (because of the AND operation with the bit value). Further, to allow non-minimal paths, the deroute ports<sup>3</sup> at switches 0 and 1 that are connected to the faulty link are also set towards the South direction ( $DR^{(0,1)} = S$ ). The rest of the failure cases present at the boundary of the  $3 \times 3$  mesh are shown in Figures 8 (b), (c), and (d). The following changes are made:

<sup>2</sup> We have assumed that generation of a new set of bits is made offline. Our aim is to offer a routing framework that can be adapted to the changes.

<sup>3</sup> Deroute ( $DR$ ) at any switch define the deadlock-free port direction. This port direction should be used for non-minimal paths.



**Fig. 8** Single link failures present at the boundary and interior links of the mesh

- To handle a failure at the lower boundary of a mesh, switch 5 resets its  $F_{sw}^5$  and  $R_{sw}^5$  bits to zero.  $F_{sw}^5$  and  $R_{sw}^5$  bits capture the fault information present at the West port of the southern neighbour (switch 8) of 5 as shown in Figure 8 (b).
- Figure 8 (c) shows a failure at the left boundary of the mesh. Switch 1 avoids the failure by resetting its  $F_{ws}^1$  and  $R_{ws}^1$  bits to zero.
- To capture a failure at the right boundary of a mesh, as shown in Figure 8 (d), switch 7 resets its  $F_{en}^7$  and  $R_{en}^7$  bits to zero.

Figures 8 (e) and (f) show a single link failure at the interior link of a switch. As shown in Figure 8 (e), to capture the failure at the South port of a Western neighbour, switch 5 resets  $F_{ws}^5$  to zero. However, in this case, 5 does not reset  $R_{ws}^5$  to zero. This is because the routing turn coded in routing bit  $R_{ws}^5$  allows a path for distant switches (from 5  $\rightarrow$  6). A similar observation can be found for the single link failure shown in Figure 8 (f).

### 3.5.2 Double Link Failure

In this section, we will discuss irregular topologies derived from two link failures in a regular 2D mesh. Different failure locations in the mesh will generate a large number of irregular topologies. For example, from a  $4 \times 4$  ( $3 \times 3$ ) mesh having 24 (12) links, a total of 276 (66) irregular topologies can be generated. Figure 9 shows few cases of  $3 \times 3$  irregular mesh topologies generated due to two link failures in the regular  $3 \times 3$  mesh. As with the single link failures discussed in the previous section, the proposed approach handles the topologies derived from double-link failures.

Figure 9 (a) shows a  $3 \times 3$  irregular mesh topology with two link failures. The link failure at the upper side boundary of the mesh is located between East of switch 0 and West of switch 1. The other failure is located between East of switch 4 and West of switch 5. The connectivity bits of the switch output ports connecting to the broken links are set to zero ( $C_e^0, C_w^1, C_e^4, C_w^5$ ). To avoid the failure at the mesh boundary, switch 3 resets  $F_{ne}^3$  and  $R_{ne}^3$  to zero. This reset prevents switch 3 considering the North port

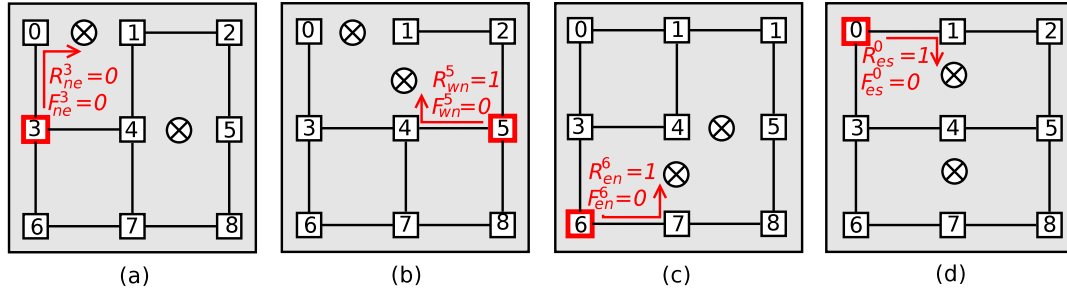


Fig. 9 Illustrative examples of irregular topologies having double-link failures

for any destinations in the  $NE$  quadrant, such as switches 1 and 2. Similarly, switch 4 also resets  $F_{nw}^4$  and  $R_{nw}^4$  to avoid the North port for the destination switch 1.

However, as shown in Figure 9 (a), the other link failure between switches 4 and 5 is interior to the mesh. Thus, switches 1, 2, 7, 8 reset their faulty bits ( $F_{sw}^2, F_{se}^1, F_{ne}^7, F_{nw}^8$ ) to zero, which in turn discards the path to neighbouring switches and allows paths to distant switches. For example, having  $F_{sw}^2 = 0$  and  $R_{sw}^2 = 1$  allows switch 2 to discard its South port for destination switch 4, and to use for switch 7. Further, to allow non-minimal paths, the deroute port at each switch with a faulty link is also set ( $DR^{(0,1)} = S, DR^{(4,5)} = N$ ).

Figures 9 (b), (c) and (d) show the cases in which both failures are present in the same switch. As explained above, each switch updates its configuration bits (routing, connectivity and faulty) to capture the failure and current topology. In the case where more than one routing option is available at any switch, Algorithm 2 is used to select the one that is most likely to be fault-free. For example, as shown in Figure 9 (b), at switch 5 routing logic (Algorithm 1) generates both  $N$  and  $W$  port directions for destination switch 0 because both  $R_{nw}^5$  and  $R_{wn}^5$  are equal to one. However, the distance of the destination switch in the  $W$  direction is larger than in the  $N$  direction and  $N$  may lead to a faulty path. As a result,  $W$  is preferred over  $N$  and the fault is avoided.

### 3.5.3 Multiple Link Failure

To adapt the proposed mechanism for multiple failures, we have classified multiple failures into two types. First, there are failures that are located at individual switches situated either far from each other or inside a mesh. Second, there are failures that are present at boundary switches and are located very close to each other. Multiple failures of the first category are easily handled by the proposed mechanism in a similar way to single and double failures, as discussed in Sections 3.5.1, 3.5.2. For example, Figure 10 (a) shows an irregular topology having a total of four failures. These failures are located at different switches of the mesh and are not concentrated at neighbouring switches. This allows faults to be considered as single link failures and can be handled in a similar way.

If the failures are very close to each other or located at boundary switches, our method disables communication between the source-destination pairs with faulty links, creating partially functioning switches. For example in Figure 10, because of the three link failures present at the boundary switches, from switch 7 to switch 1, all the possible paths in the West direction become faulty. Although, from switch 7 a single path in the North direction is available for switch 1, it is discarded by the selection logic because going North reduces the distance in a particular direction to zero. Hence, to avoid this, communication between switches 7 and 1 is disabled by the proposed mechanism.

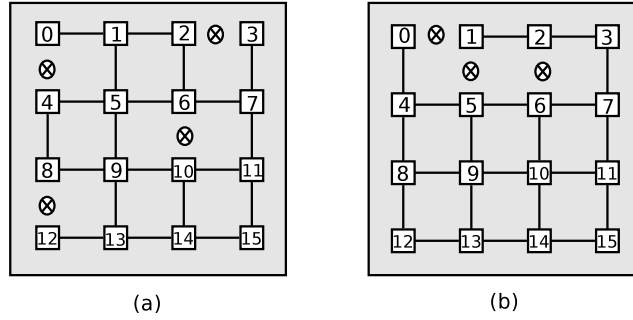


Fig. 10 Multiple link failure present at boundary switches

### 3.6 Deadlock and Livelock Freedom

The proposed approach is a routing mechanism that provides a framework to implement the rules of a routing algorithm to generate valid output ports for a packet. The underlying routing algorithm forbids some turns in order to prevent cycles and to make the CDG (Channel Dependency Graph) acyclic. We have considered the SR routing algorithm, a topology-agnostic routing algorithm. SR defines various rules for placing bidirectional routing restrictions in order to prevent cycles.

As the deadlock-freedom of SR is already proved [24], it is now the responsibility of the proposed mechanism to maintain this deadlock freedom. This means it should not generate any invalid output port that results in a violation of a routing turn prohibited by the underlying routing algorithm. As described in Section 3.2, the proposed method encodes the allowed and restricted routing options into a set of routing bits. In order to make sure that no packet will violate any forbidden turn, it sets the routing for the prohibited directions to zero and the routing bits corresponding to allowed routing options are set to one. This fact ensures that the ports corresponding to restricted turns will be discarded by the underlying routing logic (due to the AND operation with the routing bit).

Ensuring that no packet will ever cross the restricted routing option of the routing algorithm, CDG remains acyclic, making the proposed mechanism deadlock-free. Livelock is prevented by allowing only the minimal paths in the fault-free case. Non-minimal paths are allowed only when a fault is present. In this case, the particular output port (deroute port) leading to a non-minimal path towards the destination is encoded only by prior searching of the deadlock-free non-minimal path.

## 4 Experimental Evaluation and Results Analysis

We first start with an area and power analysis of a suitable router required for the proposed implementation. After that, we provide coverage analysis for various irregular topologies generated due to link failures. We also evaluate the performance with uniform, bit-reversal and bit-complement traffic profiles.

### 4.1 Area and Power Analysis

To evaluate the area overhead and power consumption of our proposed implementation compared to other logic based implementations, we have synthesized a 5-port router model for LBDR, LBDR<sub>dr</sub>, uLBDR using the Synopsys Design Compiler. For synthesis, we have used a UMC 65nm technology with a supply voltage of 1V. We have considered LBDR, LBDR<sub>dr</sub>, uLBDR for comparison with the proposed implementation because all these approaches are logic based and do not require any routing tables.



**Table 2** Area/Power Analysis

ROUTER	Area( $\mu m^2$ )	Power (mW)
		Dynamic & static
LBDR	25895	.0165
LBDR <sub>dr</sub>	37335	.0225
<b>Proposed</b>	<b>40975</b>	<b>.0237</b>
uLBDR	47319	.0281

In terms of fault tolerance, uLBDR is the most effective variant of the LBDR family of implementations. It can be observed from Table 2 that the proposed mechanism requires less area and power than the uLBDR mechanism. In order to provide 100% coverage, uLBDR imposes various major design constraints on a switch such as customized arbiter and use of VCT switching. These constraints contribute to an increase in uLBDR's overall area and power. The proposed mechanism does away with the needs of a particular switching technique and any specialized arbiter. Area and power overheads are slightly larger than for the other LBDR methods (LBDR and LBDR<sub>dr</sub>) [9]. The extra overhead of our method is coupled with improved fault coverage in comparison to other methods.

#### 4.2 Coverage Analysis

In this section, we evaluate the coverage offered by different implementations (LBDR, LBDR<sub>dr</sub>, uLBDR, and proposed). Coverage in any routing implementation is measured as the percentage of supported irregular 2D mesh topologies out of all irregular topologies that can be generated from a regular 2D mesh due to link failures. To offer resilience, 100% coverage is preferred because a single irregular topology that is not supported may ruin the entire network service [9]. A topology is considered supported if the implementation can offer connectivity between all pairs of switches while maintaining deadlock-freedom. To estimate coverage, we generated a set of irregular topologies from a regular topology by considering either 1-link or 2-link failures. For a  $n \times n$  topology we can define the following:

- The total number of links  $L = 2n(n - 1)$
- The total number of irregular topologies generated from 1-link failures =  ${}^L C_1$
- The total number of irregular topologies generated from 2-link failures =  ${}^L C_2$

For example, from an initial  $8 \times 8$  regular topology with 112 links, we can generate a set of 112 and 6216 irregular topologies having one and two failed links. To test the coverage, we have used a checker tool to check the connectivity and deadlock freedom of each implementation. The checker tool takes as input one irregular topology at a time and then performs routing using each of the methods (LBDR, LBDR<sub>dr</sub>, uLBDR, and proposed). Along with the connectivity, it also checks each method for deadlock-freedom.

For any size of the mesh, all of the mechanisms except LBDR can handle all irregular topologies generated as a result of a single link failure. A single link failure in the topology may disable the available minimal paths between a few switches, especially those situated in the same row or column of failed link. LBDR has no support for non-minimal paths, hence has zero coverage. In contrast, all three mechanisms LBDR<sub>dr</sub>, uLBDR and the proposed have support for non-minimal paths and use reconfiguration to achieve complete coverage.

Table 3 shows the coverage of different mechanisms for irregular topologies generated as a result of two link failures. We can observe that LBDR is not able to achieve complete coverage for all sizes of mesh, except for the  $2 \times 2$  mesh. It only achieves complete coverage for the  $2 \times 2$  mesh because the mesh size is too small: failures of any two links at the same time either divide the topology into two halves or convert the topology into a smaller topology having only minimal paths.

The other variant of LBDR, LBDR<sub>dr</sub>, manages to achieve good coverage, but not 100%. As mentioned above, a single unsupported irregular topology may ruin the entire network service. As an example, for an

**Table 3** Coverage for topologies having 2-link faults (LBDR<sub>dr</sub> shows the no. of topologies not supported along with percentage)

Topology size	Irregular topologies	LBDR	LBDR <sub>dr</sub>	uLBDR	Proposed
2x2	6	100%	0 (100%)	100%	100%
4x4	276	0%	16 (94%)	100%	100%
5x5	780	0%	24 (96%)	100%	100%
6x6	1770	0%	32 (98%)	100%	100%
7x7	3486	0%	40 (98%)	100%	100%
8x8	6216	0%	48 (99%)	100%	100%

8 × 8 mesh there are 48 irregular topologies that are not supported by LBDR<sub>dr</sub>. In contrast, the proposed method and uLBDR provide 100% coverage. However, to achieve 100% coverage uLBDR poses several restrictions on the switch design (a specialized arbiter for message replication and VCT switching for deadlock-freedom) and also loads the communication traffic by replicating messages.

On the other hand, our proposed mechanism does not pose any restrictions on switch design and provides the flexibility to work with any switching technique (wormhole or VCT). There is no need to replicate extra messages. Similarly, irregular topologies having multiple link failures are also handled. As explained in Section 3.5.3, if multiple link failures are present inside the boundary of the mesh and are located far apart, the proposed mechanism handles them successfully. However, if failures are present at the boundary switches and are very close to each other, it handles them by creating partial switches. So the coverage slightly reduces for topologies having more than two failures at nearby switches.

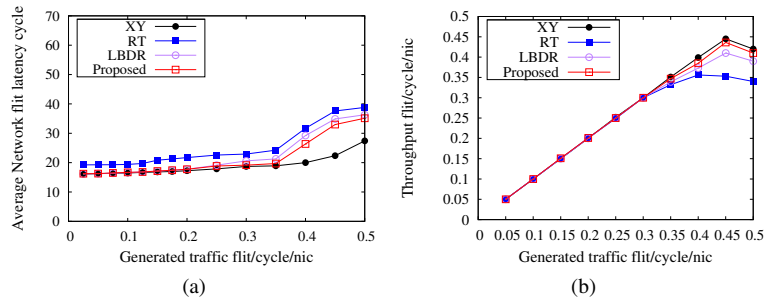
#### 4.3 Performance Analysis

To evaluate the performance of the proposed mechanism, we have used gMemNoCsim, a cycle-accurate NoC and cache hierarchy simulator developed by the Parallel Architecture Group at the Universitat Politècnica de València [25]. Average latency and throughput are used as performance metrics.

We have used a synthetic workload with the following traffic patterns: uniform, bit-reversal and bit-complement and performance values are taken at various simulation points, each with a different traffic injection rate. Stop & go flow control is used along with wormhole switching. Also to analyse the mutual effects of messages with different lengths, we have considered a weighted mix of short and long messages. Both short and long messages are of fixed sizes. To compare and analyse the performance of our proposed implementation with other similar implementation approaches, we have conducted the following two experiments.

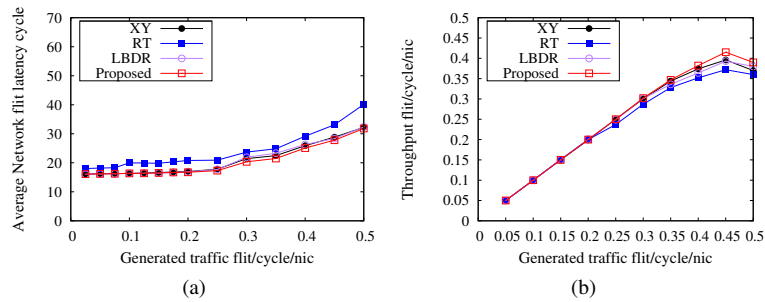
**Experiment-1** : In the first set of experiments, we have compared our proposed work with other non-fault tolerant and fault tolerant routing implementations. For this, we have considered the finite state machine based implementation of  $XY$  routing algorithm as a non-fault tolerant implementation. For fault-tolerant implementations, we have considered Routing Tables (RT), LBDR and our proposed method along with the segment-based routing algorithm. For this experiment, 40K permanent and 20K transient messages are generated during a single simulation point. The message size is set to 3 and 10 flits, and the buffer size is set to 8 flits.

Figures 11 (a) and (b) show the average network latency and throughput under the uniform distribution of messages. We can observe that under the uniform traffic profile  $XY$  routing algorithm outperforms all other mechanisms by achieving lower latency and higher throughput because the network is uniformly loaded.  $XY$  routing is not fault-tolerant and will not work for many source-destination pairs even in the case of a single link failure. The routing table (RT) based implementation, although flexible in handling each type of failure, is more expensive than all other mechanisms in terms of both space and time. RT results in higher latency and lower throughput than the other mechanisms, mainly due to its additional table access latency. We can also observe that LBDR exhibits an improvement in performance over the

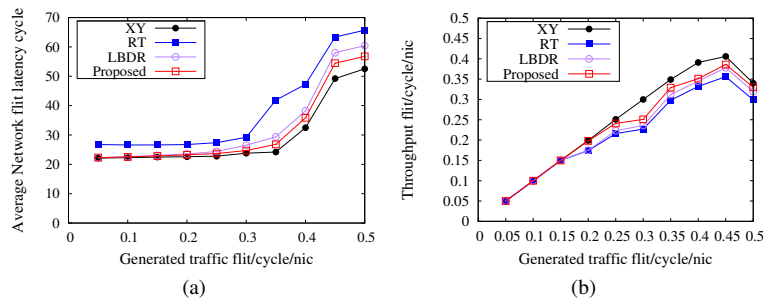


**Fig. 11** Comparison of different routing implementations with uniform traffic pattern

RT-based implementation due to its simplified routing logic design. With low injection rates, LBDR and the proposed work exhibit similar performance but our proposal scores over LBDR at high injection rates because of its selection strategy that intelligently distributes the traffic over other available links.



**Fig. 12** Comparison of different routing implementations with bit-reversal traffic pattern



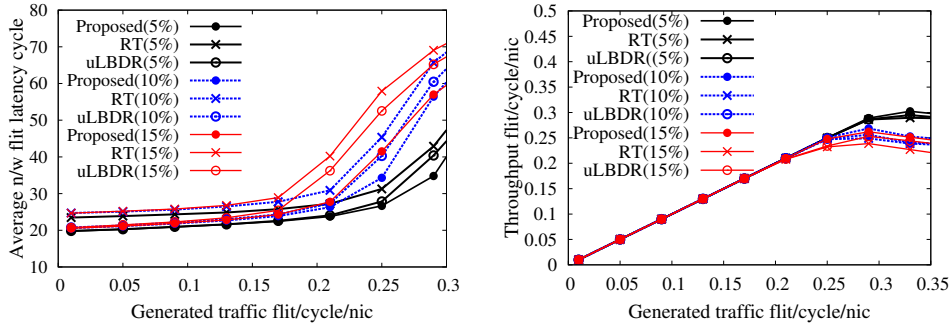
**Fig. 13** Comparison of different routing implementations with bit-complement traffic pattern

Figures 12 (a) and (b) show the average network latency and throughput with the bit-reversal pattern. In terms of latency and throughput, LBDR and our proposed method exhibit similar performance to XY at low injection rates. At high injection rates, our proposed mechanism outperforms the others. RT results in higher latency and lower throughput under the bit-reversal traffic pattern also. The observed performance degradation of RT may be attributed to delays because of table access.

Figures 13 (a) and (b) show the average network latency and throughput according to the bit-complement pattern. The trends observed here are quite similar those for uniform traffic distribution, the differences being that their performance degradation onsets are at lower injection rates. The latency of RT starts saturating at an injection rate of 0.3, whereas the proposed method, LBDR and  $XY$  saturate at higher injection rates near to 0.4, as shown in Figure 13 (a). For throughput, all mechanisms start saturating at around 0.45 injection rate as shown in Figure 13 (b). The saturation is due to back pressure as an NoC cannot handle this or higher injection rates.

**Experiment-2:** In Experiment-1, the performance of our proposed work was evaluated by considering a regular 2D mesh topology during each simulation. However, the performance of any fault tolerant routing framework is also important when the topology becomes irregular due to the presence of failures. In this set of experiments, we have compared the performance of the proposed mechanism with other fault tolerant implementations (RT, uLBDR) under failure conditions. Unlike Experiment-1,  $XY$  is not considered in this case as it does not support failures. For comparison, we have considered irregular 2D mesh topologies generated due to failures in a regular 2D mesh. The purpose is to demonstrate that under the irregular topological scenario, the proposed implementation does not suffer degraded performance instantly.

Each simulation is run with a different set of irregular 2D mesh topologies derived from a regular 2D mesh by injecting link faults with incidences of 5, 10, and 15 percent of the total links. More specifically, for a varying set of fault rates we have observed the latency and throughput parameters of the proposed approach. For this experiment, instead of a wormhole, virtual-cut-through (VCT) switching is used. As explained in the introduction section, uLBDR is constrained to work with only VCT switching [9]. During each simulation point, 20K permanent and 10K transient messages are generated. As in VCT switching, message size should be equal to the buffer size, both are set to 8 flits.



**Fig. 14** Performance comparison of different mechanisms during failures under uniform traffic profile

Figures 14 (a) and (b) show the average network latency and throughput of different mechanisms as a function of the injection rate on a 2D mesh when using uniform traffic distribution. We can observe that for each mechanism, in the case of failures, the latency is slightly higher than the case with no failures (Experiment-1). However, as in Experiment-1, for each fault rate, the proposed approach maintains lower latency as compared to other approaches. The reason is that, in the event of a failure, it first tries to find an alternative fault-free minimal path; otherwise it opts for a non-minimal path in some cases. In cases where the minimal path is not available, a non-minimal path needs to be taken. This fact slightly lowers the throughput for all approaches.

A similar observation can be seen in the latency and throughput graphs of bit-reversal (Figure 15) and bit-complement (Figure 16) traffic patterns. In the case of failures, as compared to other approaches, the proposed approach gracefully degrades its performance while preserving the connectivity between each

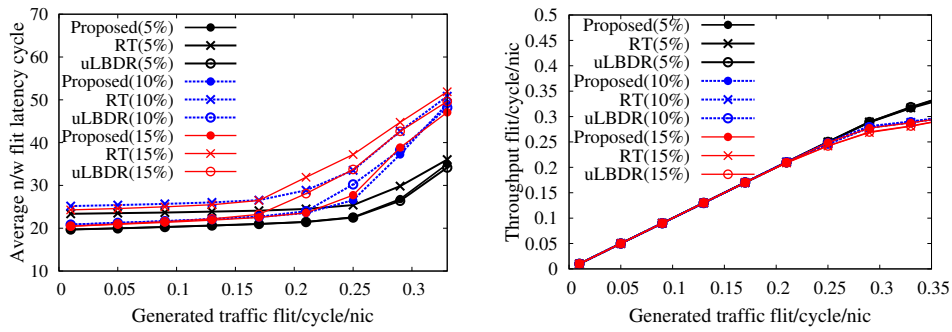


Fig. 15 Performance comparison of different mechanisms during failures under bit-reversal traffic profile

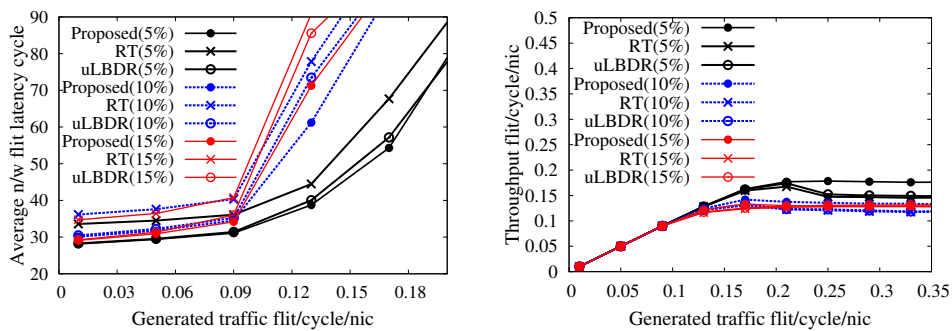


Fig. 16 Performance comparison of different mechanisms during failures under bit-complement traffic profile

source-destination pair. It is important to notice that this degradation in performance is mainly due to the presence of failures affecting the total available paths between source-destination nodes. As the topology changes from regular to irregular, a few links become unavailable due to the faults. This forces some packets to take alternative minimal paths or non-minimal paths if available. In the failure case, any of the resultant paths add some performance overhead due to congestion or the length of the path.

## 5 Conclusions

To deal with the challenges of on-chip routing, we have proposed a novel, fault-resilient routing method. The proposed work handles failure-induced irregular topologies derived from regular 2D meshes. It incorporates segment-based routing, a topology-agnostic routing algorithm as the underlying routing algorithm. The proposed mechanism is based on a few configuration bits per switch and captures the routing algorithm and topology. The number of configuration bits does not increase with network size, hence leads to a scalable design, unlike routing table based implementations. We demonstrate that our proposed method is guaranteed to tolerate all locations of single and double-link failures and most multiple failures. Unlike uLBDR, the proposed mechanism can be configured under both wormhole and virtual-cut-through switching and does not replicate any extra messages. Along with fault tolerance, our proposed mechanism also gives priority to performance. The proposed method can achieve improved network performance for fault-free cases while achieving graceful performance degradation during failure. The proposed mechanism handles irregularities with a 14% reduction in area requirements and a 16% reduction in overall

power consumption when compared with other published work, and thus, provides a promising solution for future many-core architectures.

**Acknowledgements** We would like to acknowledge support received from Indo-Spain DST project vide contract number DST/INT/Spain/P35/11/1 and UK India Education and Research Initiative (UKIERI - II) funded project "HiPER NIRGAM" vide contract number IND/Cont/E/11-12/78.

## References

1. J. Duato, S. Yalamanchili, and L. Ni. *Interconnection Networks: An Engineering Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002.
2. S. Borkar. Microarchitecture and design challenges for gigascale integration. In *37th International Symposium on Microarchitecture, 2004. MICRO-37 2004.*, pages 3–3, Dec 2004.
3. C. Constantinescu. Trends and challenges in VLSI circuit reliability. *IEEE, Micro*, 23(4):14–19, July 2003.
4. G. De Micheli and L. Benini. *Networks on chips: technology and tools*. Academic Press, 2006.
5. T. Skeie, F.O. Sem-Jacobsen, S. Rodrigo, J. Flich, D. Bertozzi, and S. Medardoni. Flexible DOR routing for virtualization of multicore chips. In *Proc SOC'09*, pages 073–076, Oct 2009.
6. Y. B. Kim and Y.-B. Kim. Fault tolerant source routing for network-on-chip. In *Proc. DFT '07*, pages 12–20, 2007.
7. 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 '09.*, pages 21–26, April 2009.
8. C. Feng, Z. Lu, A. Jantsch, M. Zhang, and Z. Xing. Addressing transient and permanent faults in NoC with efficient fault-tolerant deflection router. *IEEE Transactions on VLSI Systems*, 21(6):1053–1066, June 2013.
9. S. Rodrigo, J. Flich, A. Roca, S. Medardoni, D. Bertozzi, J. Camacho, F. Silla, and J. Duato. Cost-efficient on-chip routing implementations for CMP and MPSoC systems. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, 30(4):534–547, April 2011.
10. D.H. Linder and J.C. Harden. An adaptive and fault tolerant wormhole routing strategy for k-ary n-cubes. *IEEE Transactions on Computers.*, 40(1):2–12, Jan 1991.
11. W. J. Dally, L. R. Dennison, D. Harris, K. Kan, and T. Xanthopoulos. The reliable router: A reliable and high-performance communication substrate for parallel computers. In *Proceedings of the First International Workshop on Parallel Computer Routing and Communication*, PCRCW '94, pages 241–255, London, UK, 1994. Springer-Verlag.
12. M.E. Gomez, J. Duato, J. Flich, P. Lopez, A. Robles, N.A. Nordbotten, O. Lysne, and T. Skeie. An efficient fault-tolerant routing methodology for meshes and tori. *Computer Architecture Letters*, 3(1):3–3, January 2004.
13. C.-T. Ho and L. Stockmeyer. A new approach to fault-tolerant wormhole routing for mesh-connected parallel computers. *IEEE Transactions on Computers.*, 53(4):427–438, April 2004.
14. C.J. Glass and L.M. Ni. Fault-tolerant wormhole routing in meshes. In *Proc FTCS-23'93*, pages 240–249, June 1993.
15. T. Schonwald, J. Zimmermann, O. Bringmann, and W. Rosenstiel. Fully adaptive fault-tolerant routing algorithm for network-on-chip architectures. In *10th Euromicro Conference on Digital System Design Architectures, Methods and Tools, 2007(DSD 2007).*, pages 527–534, Aug 2007.
16. V. Puente, J.A. Gregorio, F. Vallejo, and R. Bevide. Immunit: a cheap and robust fault-tolerant packet routing mechanism. In *Computer Architecture, 2004. Proceedings. 31st Annual International Symposium on*, pages 198–209, June 2004.
17. D. Fick, A. DeOrio, Jin Hu, V. Bertacco, D. Blaauw, and D. Sylvester. Vicis: A reliable network for unreliable silicon. In *46th ACM/IEEE Design Automation Conference, 2009(DAC '09)*, pages 812–817, July 2009.
18. K. Aisopos, A. DeOrio, Li-Shiuan Peh, and V. Bertacco. Ariadne: Agnostic reconfiguration in a disconnected network environment. In *International Conference on Parallel Architectures and Compilation Techniques(PACT), 2011*, pages 298–309, Oct 2011.
19. S. Rodrigo, S. Medardoni, J. Flich, D. Bertozzi, and J. Duato. Efficient implementation of distributed routing algorithms for NoCs. *Computers Digital Techniques, IET*, 3(5):460–475, 2009.
20. A. Ghiribaldi, D. Ludovici, F. Triviño, A. Strano, J. Flich, J. L. Sánchez, F. Alfaro, M. Favalli, and D. Bertozzi. A complete self-testing and self-configuring noc infrastructure for cost-effective mpsoes. *ACM TECS*, 12(4):106:1–106:29, 2013.
21. A. Mejia. *Design and Implementation of Efficient Topology Agnostic Routing Algorithms for Interconnection Networks*. PhD thesis, University of Valencia, 2008.
22. J. Flich, T. Skeie, A. Mejia, O. Lysne, P. Lopez, A. Robles, J. Duato, M. Koibuchi, T. Rokicki, and J.C. Sancho. A survey and evaluation of topology-agnostic deterministic routing algorithms. *IEEE Transactions on Parallel and Distributed Systems*, 23(3):405–425, March 2012.
23. M.D. Schroeder, A.D. Birrell, M. Burrows, H. Murray, R.M. Needham, T.L. Rodeheffer, E.H. Satterthwaite, and C.P. Thacker. Autonet: a high-speed, self-configuring local area network using point-to-point links. *IEEE Journal on Selected Areas in Communications.*, 9(8):1318–1335, Oct 1991.
24. A. Mejia, J. Flich, and J. Duato. On the potentials of segment-based routing for NoCs. In *Proc. ICPP'08*, pages 594–603, 2008.
25. gmemocsim. Available at. <http://www.gap.upv.es>, 2010.