IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 8, 2022, accepted April 10, 2022, date of publication April 20, 2022, date of current version May 2, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3168992

Reinforcement Learning Based Fault-Tolerant Routing Algorithm for Mesh Based NoC and Its FPGA Implementation

SAMALA JAGADHEESH[1], (Graduate Student Member, IEEE), P. VEDA BHANU[1], J. SOUMYA[1], AND LINGA REDDY CENKERAMADDI[2], (Senior Member, IEEE)

[1]Department of Electrical and Electronics Engineering, Birla Institute of Technology and Science-Pilani, Hyderabad Campus, Hyderabad, Telangana 500078, India

[2]Department of Information and Communication Technology, University of Agder (UiA), Grimstad, 4630 Kristiansand, Norway

Corresponding author: Linga Reddy Cenkeramaddi (linga.cenkeramaddi@uia.no)

This work was supported in part by the Indo-Norwegian Collaboration in Autonomous Cyber-Physical Systems (INCAPS) of the INTPART International Partnerships for Excellent Education, Research and Innovation Program from the Research Council of Norway under Project 287918.

ABSTRACT Network-on-Chip (NoC) has emerged as the most promising on-chip interconnection framework in Multi-Processor System-on-Chips (MPSoCs) due to its efficiency and scalability. In the deep submicron level, NoCs are vulnerable to faults, which leads to the failure of network components such as links and routers. Failures in NoC components diminish system efficiency and reliability. This paper proposes a Reinforcement Learning based Fault-Tolerant Routing (RL-FTR) algorithm to tackle the routing issues caused by link and router faults in the mesh-based NoC architecture. The efficiency of the proposed RL-FTR algorithm is examined using System-C based cycle-accurate NoC simulator. Simulations are carried out by increasing the number of links and router faults in various sizes of mesh. Followed by simulations, real-time functioning of the proposed RL-FTR algorithm is observed using the FPGA implementation. Results of the simulation and hardware shows that the proposed RL-FTR algorithm provides an optimal routing path from the source router to the destination router.

INDEX TERMS Fault-tolerance, FPGA, network-on-chip, reinforcement learning, routing.

I. INTRODUCTION

Nowadays Network-on-Chip (NoC) became popular on-chip communication paradigm for many core systems. NoCs facilitate parallel processing by providing high bandwidth and low latency, which helps to deliver high computational power for real-time as well as safety-critical applications [1]. NoCs are designed using regular or irregular topologies. Mesh is a basic regular topology formed by interconnecting the neighboring routers in a grid manner. The structure of mesh topology is simple and easy to explore.

In NoC, routers route the packets received from the source cores towards the destination cores via links. The routing algorithm embedded inside the router is responsible for forwarding the packets towards destination, which plays a vital role in successful delivery of packets [2]. Miniaturization of transistor and technology scaling help to integrate more number of transistors in a small chip area. NoC serves the requirements of more processing elements (PE) or cores, resulting in high switching activity, and heat dissipation. As a consequence, network components are likely to malfunction [3]. Failure of the network components adversely affects the performance and reliability of the system. So, to improve reliability and efficiency, NoCs require fault-tolerant approaches to tackle faults.

In general, conventional fault-tolerant routing algorithms (or) shortest path routing algorithms make decisions based on the predefined rules. Also, the packets always go through the same node in the path to the destination because of lack of intelligence, which creates congestion and queuing problems. The rules are user-defined based on frequently occurred routing problems observed by the programmer, and on every new scenario, human intervention is required to update the rules [4]. However, as the number of routing problems increases, demand to define new rules increases to accurately address all routing problems, resulting in loss of efficiency or accuracy. In Machine Learning (ML), algorithms are programmed to learn to perform the

The associate editor coordinating the review of this manuscript and approving it for publication was Nitin Nitin.

task [5]. During the learning process, the ML algorithm acquires knowledge of different routing scenarios, which helps to handle complex situations efficiently and accurately. This motivated us to work on ML to propose a fault-tolerant routing algorithm for mesh-based NoC.

Machine Learning (ML) is one of the most demanding techniques in the current market. It has the ability to perform intelligent tasks such as classify, recognize, advise, optimize, and predict. RL is one of the ML algorithms, it is a goal-oriented learning based on the interaction with the environment. The environment is a set of states or tasks. In RL, the agent takes the actions in order to minimize or maximize the cumulative reward depending on the reward policy [6]. The cumulative reward helps to find the solution. In this paper, Reinforcement Learning based Fault-Tolerant Routing (RL-FTR) algorithm is proposed to address the link as well as router faults present in mesh topology based NoC. The proposed RL-FTR algorithm uses multi-agent reinforcement learning (MARL) algorithm to find the optimal routing path between the source router and destination router. MARL is the area that focuses on the implementation of autonomous, self-learning systems with multiple agents. Conceptually, MARL is a deep learning discipline that focuses on models, which include multiple agents that learn by dynamically interacting with their environment.

The proposed RL-FTR algorithm is tested on software and hardware platforms to observe its functionality and efficiency in-terms of latency and packet delivery. As part of software platform testing, the proposed RL-FTR algorithm functionality is tested by implementing in System-C based cycle-accurate NoC simulator. As a part of hardware testing, the real-time functionality of the proposed RL-FTR algorithm is observed using the FPGA-based NoC prototype. FPGA implementation helps to identify and solve the timing and functional issues and also, reduces the pre-silicon design verification time. Significant contribution of this paper are listed below:

1) RL based fault-tolerant routing algorithm is proposed to tackle the links and routers faults in mesh based NoC.
2) The proposed RL-FTR algorithm is implemented in a System-C based NoC simulator, and a detailed analysis of average network latency and packet loss is reported.
3) Real-time behavior of the proposed RL-FTR algorithm is observed using the case studies by implementing it on FPGA.
4) Resource utilization and power analysis are reported for the proposed RL-FTR algorithm and compared it with the algorithms proposed in [7], [8] and [9].

The structure of the paper is as follows: A brief literature review on related works is reported in Section II. Outline of the mesh topology and reinforcement learning are described in Section III. In Section IV, formulations for the fault-tolerant routing algorithm are discussed. Simulation results analysis and FPGA implementation with case studies of the proposed RL-FTR algorithm are discussed in Section V. Section VI concludes the paper.

II. RELATED WORK

Many researchers proposed fault-free and fault-tolerant routing algorithms for mesh topology based NoCs. In [10], authors proposed a fault-tolerant routing algorithm using virtual channels (VCs). The algorithm requires a knowledge base to get fault-related information, which increases processing overhead. The authors in [11], developed a reconfigurable router architecture called DSPIN (Distributed Scalable Predictable Interconnect Network) and a reconfigurable routing algorithm for the developed architecture. The DSPIN router architecture has VCs as well as turn around model, which increases the area overhead. The authors in [12], proposed a MinFT routing algorithm for mesh topology, which is partially fault-tolerant. In [13], the authors proposed Improved-Fault-Tolerant-Algorithm (i-FTR), it uses VCs to pass the faulty region and provides dead-lock free routing. A fault-tolerant NoC architecture is proposed [14] by adding spare links and control units to mesh topology. The proposed architecture has a new routing algorithm that accesses the required information from control units to provide routing paths and works only for faulty routers. The authors in [9], developed a dynamic fault-tolerant XY-YX routing algorithm. In fault-free situations, it serves as a traditional XY routing algorithm, but it switches from XY to YX routing in the event of failure. Further in [15], the authors added extra switches and links to the router architecture. Switches at the failed node transfer the data to spare links. Additional switches and links improve the latency while increasing hardware overhead. The authors in [16], introduced a new concept un-routing in the proposed Dn-FTR routing algorithm. Un-routing allows the packet to traverse back to the previous node whenever the forward-path is not available. The authors in [8], proposed a routing algorithm to address link faults and modified the packet header to include fault related information. The packet header has a 2-bit field to define the fault direction. Based on the fault information, the packet is routed towards the destination.

Q-routing based self-regulated routing algorithm for NoC is proposed in [17]. Based on the congestion, the proposed algorithm dynamically changes the NoC routing scheme to improve the packet latency. In [18], authors proposed RL based routing for adaptive traffic optimization to improve the performance of NoC. In [19], the authors proposed a control policy using RL to enhance the performance of NoC. The proposed policy optimally operates the error detection, error correction and re-transmission of the packet to reduce power consumption and latency. RL based control policy is proposed in [20] to improve the energy efficiency of NoC by observing and optimizing the usage of different components such as cache and buffers. The authors in [21], proposed an Intelligent NoC design framework (IntelliNoC) using RL. It manages the complexity of the design while optimizing

energy efficiency, performance, and reliability. In [22], the authors proposed a learning-based NoC design CURE. It has a reversible multi-function adaptive channel, enhanced fault-tolerant router circuitry, ten unique operation modes, and a Deep Reinforcement Learning (DRL) based dynamic control policy. DRL-based control policy acquires NoC behavior knowledge and operates it in optimal mode to improve energy efficiency, performance, and reliability. A Q-learning based fault-tolerant routing algorithm is proposed in [7]. The Q-learning has Q-table, which helps to take the routing path related decision. In practical, for the systems with high number of routers, requires more memory to accommodate Q-table and also the learning agent require more iterations to explore the network.

In the literature, most of the works are addressing a limited number of faulty routers and links in NoC. In some of the literature works, RL-based techniques have been used to improve the NoC design, to enhance the energy efficiency, performance, and reliability of the NoC. Previous research works show that RL has the potential to tackle the routing problems in NoC. This motivated us to use RL techniques to propose an efficient fault-tolerant routing algorithm for mesh topology based NoC. The proposed Reinforcement Learning based Fault-Tolerant Routing (RL-FTR) uses decentralized MARL [23] with networked agents. In decentralized MARL, each agent makes its own decision, based on only local observations and information transmitted from its neighbors, and without coordination by a central controller. This helps multiple agents perform sequential decision-making in a common environment and improves the scalability of Q-table. The proposed Reinforcement Learning based Fault-Tolerant Routing (RL-FTR) algorithm addresses link and router faults present in mesh topology based NoC.

III. OVERVIEW OF MESH TOPOLOGY AND REINFORCEMENT LEARNING

A. MESH TOPOLOGY

NoC designs have regular and irregular (application-specific) topologies. Mesh is considered as one of the regular topologies in NoC. In general, the size of mesh topology is represented as $m \times n$, where m and n indicate the number of rows and columns respectively. Figure. 1 shows the mesh topology of size 4×4 along with the link directions of the router. Each router in the mesh topology can facilitate one core. Parameters of $m \times n$ mesh topology are as follows:

- Number of routers required: (mn)
- Number of links: $(m\,(n-1) + n\,(m-1))$
- Diameter: $(m + n - 2)$
- Average distance: $(m + n)\,/\,3$
- Bisection width: $\min(m, n)$
- Node degree: *3 (corner), 4 (boundary), 5 (center)*

Mesh topology can be divided into two parts by disconnecting a minimum of m or n links. The routers in the corner of mesh topology are connected with two neighbour routers and one core, boundary routers are connected with three neighbour routers and one core, and center routers are connected with four neighbour routers and one core.

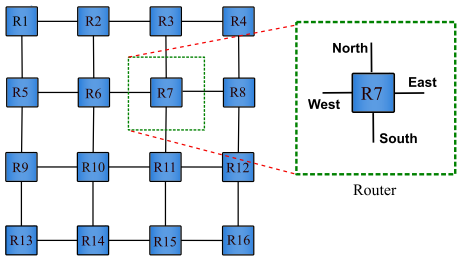


FIGURE 1. Mesh topology (4 × 4).

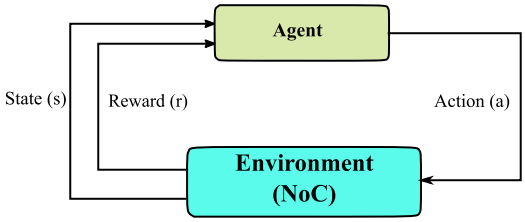


FIGURE 2. Interaction of agent with the environment in RL.

B. REINFORCEMENT LEARNING

RL is a sub-area in ML, where an agent learns to perform a task in an environment by taking sequential actions and observing the rewards received for actions [6]. Depending on the reward policy, every action gets a reward. RL agent learning is an iterative process, which updates the reward of every state-action pair. As a result of the learning process, the cumulative reward provides the optimal solution. The procedure of RL agent and environment interaction is presented in Figure. 2.

In general, RL is considered as a Markov Decision Process (MDP) [24]. MDP is a tuple denoted by (S, A, T, R, γ), where *'S'* is the set of states, *'A'* is the set of actions available at each state, *'T'* is the state transition function $T : S \times A \times S \rightarrow [0,1]$, *'R'* is the reward function provides the reward for the actions $R : S \times A \times S \rightarrow R$, and γ is the discount factor decides the significance of future rewards respectively. In RL, the environment is entirely observable, which means that the agent interacts with the environment at every action. Based on the current state and the action taken, the agent receives a reward as feedback for the action and moves to the next state.

IV. PROPOSED ROUTING ALGORITHM

Routing in mesh topology is modeled as a Markov Decision Process (MDP) [25]. In general, the RL agent learns using updating policies to find the optimal solution. Similarly, the RL agent explores the NoC environment and finds the optimal routing path between the source and destination routers. MDP for the routing in mesh topology is as follows:

- **State *(S)*:** Routers in the mesh topology are assumed as the states. In the routing path, source router, destination

router, and current router are considered as the initial state, target state, and current state, respectively.

- **Action *(A)*:** Links connected to the routers in mesh topology are assumed as the actions. Based on the position, each router have a different number of adjacent routers. Correspondingly, every state has a different number of actions.
- **Transition *(T)*:** Since the routers modeled as states, the movement from one router to its neighbour router is treated as state transition from current state to next state.
- **Reward *(R)*:** Based on the reward policy, the RL agent receives a positive or negative reward for each action performed at every state. The cumulative reward of state-action pairs (s_i,a_i) help to obtain the solution.
- **Learning rate (α):** Learning rate determines the amount of newly learned value which is used to update the old value. It varies between 0 and 1. If the learning rate is 0, the agent will not acquire any new knowledge. If the learning rate is 1, the agent will discard the old learned value and acquires only the new learned value.
- **Discount factor (γ):** The discount factor shows the importance of past, present and future rewards received by the agent at the current state. It is a real value ranges from 0 to 1. If the value of discount factor is '0', the agent takes care of the achieved reward. The discount factor value '1' will enable the agent to aim for a high long-term reward.

In the deep learning ecosystem, multi-agent reinforcement learning (MARL) is the area that focuses on the implementation of autonomous, self-learning systems with multiple agents [23]. Conceptually, MARL is the deep learning discipline that focuses on models that include multiple agents that learn by dynamically interacting with their environment. While in single-agent reinforcement learning scenarios the state of the environment changes solely as a result of the actions of an agent, in MARL scenarios the environment is subjected to the actions of all agents. Routing on large-scale interconnect networks is by nature a MARL problem because routers in the system can be considered as independent agents. In this work, we consider mesh routing problem as a cooperative independent MARL process that routers behave as independent learners for a common objective, that is, to deliver messages in the shortest path. The proposed RL-FTR has three major components: (1) two-level Q-table, (2) Q-table update, and (3) routing with Q-table.

A. TWO-LEVEL Q-TABLE

In general, Q-routing adopts table based RL for decision making. Table-based RL methods are computationally efficient as compared to deep neural network based methods, hence being more realistic for practical use. However, Q-table mitigates outdated Q-value issue commonly experienced in large-scale systems and also requires large memory to store the table. To overcome the challenges in Q-table the proposed RL-FTR utilizes a two-level Q-table. The two-level Q-table provides more learning information while using half the amount of memory as the Q-table used in Q-routing. New techniques are adopted for RL-FTR to ensure timely update of values in the two level Q-table, hence guarantee a fast and stable model convergence.

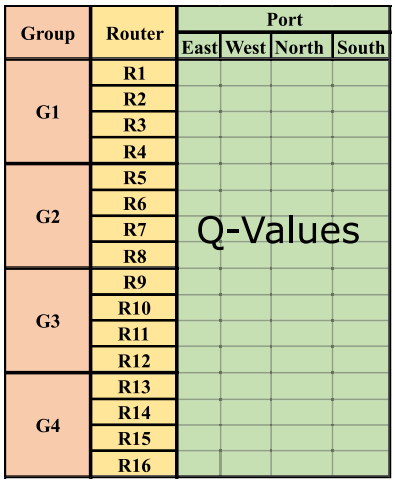


FIGURE 3. Two level Q-table in mesh topology (4 × 4).

Figure 3 shows the structure of two level Q-table in mesh of size 4 × 4. The routers in the mesh topology are grouped based on the row. The first level of the table contains the group information. The second level is the Q-values of the ports that are associated with the router.

B. Q-TABLE UPDATE

In RL, many methods are available to obtain the optimal solution. Q-learning is one of the model-free approaches in RL [26]. It is a value-based learning algorithm, which updates values based on actions of learning agent in the environment. The proposed RL-FTR algorithm used Q-learning to obtain the shortest routing path from the source to destination router in mesh topology. Q-learning has a Q-table to hold the Q-value of every state-action pair $V(s_i,a_i)$ in the environment. Q-value update policy equation for the Q-learning is as follows:

$$V(s_i, a_i) \leftarrow V(s_i, a_i) + \alpha\,[\, r_i + \gamma \max_a V(s_{i+1}, a) - V(s_i, a_i)] \tag{1}$$

In the proposed RL-FTR algorithm, the mesh topology is modeled as the environment, routers as states, and the links associated with the routers as the actions. As shown in Figure. 1, every router in mesh topology has a different number of active links depending on the location, such as corner routers have two active links, edge routers have three active links and middle routers have four active links. Similarly, each state in the Q-table has four actions. Out of four actions, few are allowed actions depending on the position of the router. Figure. 4 shows the Q-value table of RL in NoC prospect for mesh topology of size 4 × 4. In the proposed RL-FTR, every router has an individual learning agent. During the learning process of the agent begins from

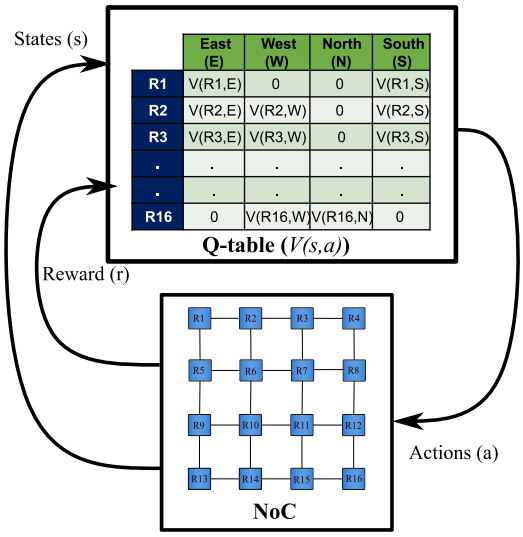


FIGURE 4. Q-values table for mesh of size 4 × 4.

the source router as an initial state (s_0). At every state (s_i), the agent performs an action, accordingly receives a reward, and makes a transition to the next state (s_{i+1}). Based on the received reward and state transition, Q-value for the current state-action pair $V(s_i,a_i)$ is updated using equation (1). The whole procedure is repeated for a limited number of iterations until the convergence of Q-table. Algorithm 1 describes the pseudo-code for the Q-learning algorithm to find the shortest routing path in mesh topology. Algorithm 1 works for mesh topology based NoC without any faults or with link and router faults because the rewards are given based on the condition of NoC.

Algorithm 1: Proposed Reinforcement Learning Based Fault-Tolerant Routing Algorithm

Input: Source router address, destination router address and mesh topology (faulty or fault-free)
Output: Optimal routing path from source router to destination router

Initialize $V(s,a) = 0$
  Initial state (s_0) = Source router address
  Target state (s_d) = Destination router address
  Current state (s_i) = Initial state;
/* Q-table updating process*/
for *epoch* $\leftarrow 1$ *to* N **do**
  if *Current state* $!=$ *Target state* **then**
    Perform a random action and update $V(s_i, a_i) \leftarrow V(s_i, a_i) + \alpha\,[\, r_i + \gamma \max_a V(s_{i+1}, a) - V(s_i, a_i)]$
    Current state $= (s_{i+1})$
  end
  else if *Current state* $=$ *Target state* **then**
    Current state = Initial state
  end
end

Reward policy for updating the Q-value is taken from the structure of mesh topology (fault-free and faulty), and every action has a predetermined reward value specified by the programmer. An action leading to the state transition, i.e., from router to router, if both are connected, will receive a reward of 100. If both routers are not connected or faulty, no reward will be given, and if the next state is the destination, it will receive a reward of 1000. Q-values are updated during the learning process based on the interaction of the agent with the environment as well as with the other agents. At the end of the learning process, Q-table has the cumulative Q-values $(V(s_i,a_i))$ for every state-action pair. In the process of finding the optimal routing path, the action with the highest Q-value is picked at the initial state, and based on the action, state transition occurs. This process is repeated at every current state until the packet reaches the destination router.

C. ROUTING WITH Q-TABLE

In the proposed RL-FTR, the router first identifies the group of the destination. Based on the group the destination is present, port with the best Q-value is selected at the source router. Depending on the selected port the data reaches to the next router and the process is continued till the data reaches the destination. During the learning process, all the agents in the RL-FTR explores the complete mesh topology.

Figure. 5 shows an example to find the shortest path between the routers R1 and R16 using the proposed RL-FTR algorithm. Here R1 is the source router and R16 is the destination router. Figure. 5a shows the updated table after the learning process. Out of all available ports at the source router R1, East port has the highest accumulated Q-value. So, the east port is selected, accordingly the data transfers to the next router, i.e., router R2. Similarly, the port selected at R2 router is west, which transfers the data to R6. This process is continued until the data reaches destination router R16. Figure. 5b shows the obtained shortest routing path between the routers R1 and R16.

Similar to the example shown in Figure. 5, for all the sources to destinations the RL-FTR algorithm has control over the complete routing path, hence the proposed RL-FTR algorithm is free from *dead-lock* and *live-lock*.

V. EXPERIMENTAL RESULTS AND ANALYSIS

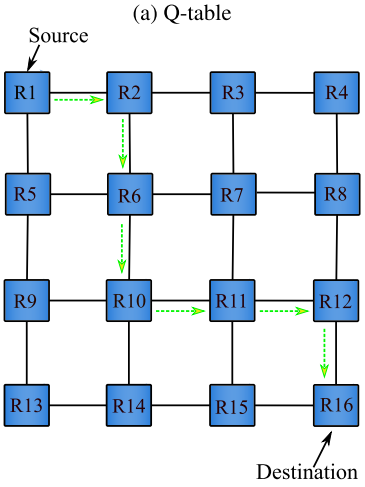
In this section, the performance of the proposed RL-FTR algorithm is observed using the NoC simulator. Besides, real-time functionality is observed using FPGA implementation.

A. IMPLEMENTATION ON NoC SIMULATOR

In this section, scalability and efficiency of the proposed RL-FTR algorithm is tested using the System-C based cycle accurate NoC simulator [27]. The simulations are performed on different sizes of mesh topology. The algorithm is tested for various conditions of NoC by increasing the number of faulty links or routers or the combination of faulty links and routers present in the mesh network.

Group	Router	Port				Path
		East	West	North	South	
G1	R1	**439.81**	0	0	431.66	R2
	R2	330.71	**616.23**	0	644.22	R6
	R3	439.01	812.33	0	826.8	
	R4	0	444.44	0	627.27	
G2	R5	317.8	0	619.58	610.97	
	R6	461.19	413.8	815.99	**861.71**	R10
	R7	616.14	619.69	1089.2	1027.2	
	R8	0	583.8	1100.7	1130.2	
G3	R9	465.65	0	812.94	820.46	
	R10	**1153.1**	615.09	644.13	1045.2	R11
	R11	**1545.6**	856.45	1128.9	817.87	R12
	R12	0	2762.3	2674.4	**3683.9**	R16
G4	R13	825.56	0	1468.1	0	
	R14	1149.7	1102	1956.1	0	
	R15	1321.7	1468.1	2762.3	0	
	R16	0	0	0	0	

(a) Q-table



(b) Obtained routing path from R1 to R16

FIGURE 5. Example: Routing path finding between R1 and R16 in mesh topology (4 × 4) using the proposed algorithm.

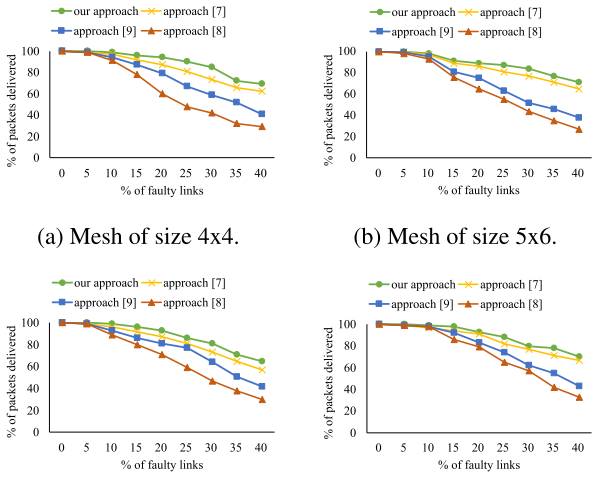
TABLE 1. NoC simulator configuration parameters.

Parameter	Value
Traffic	Synthetic
Injection rate	0.02 Packets/Cycle/IP
Simulation time	2,00,000 Cycles
Saturation time	10,000 Cycles
Switching type	Wormhole
Flit length	32 bits
Packet size	64 flits (1 Header, 62 Payload, 1 Tailer)

1) EXPERIMENTAL SETUP

The proposed RL-FTR algorithm is implemented in System-C language and integrated with the System-C based cycle-accurate NoC simulator [27]. The routing algorithm takes the connection file as the input during the initial setup process. It updates the Q-tables, which contains information about routing in mesh topology. The fault related information is provided to the routing algorithm through the connection file.

Table 1 shows the NoC simulator configuration parameters used for all the experimentations. The simulations are performed on a machine having Intel Xeon E5-1650 v3 processor and 32 GB RAM.



(a) Mesh of size 4x4. (b) Mesh of size 5x6.

(c) Mesh of size 6x7. (d) Mesh of size 8x8.

FIGURE 6. Change in percentage of packets delivered with respect to change in the percentage of faulty links.

2) EXPERIMENTAL RESULTS ANALYSIS

The simulations are performed on different dimensions of mesh topology by increasing the number of faults present in mesh topology. Based on the kind of faults present in the mesh topology, the simulations are performed in three groups. Group-wise analysis of the simulation results is as follows:

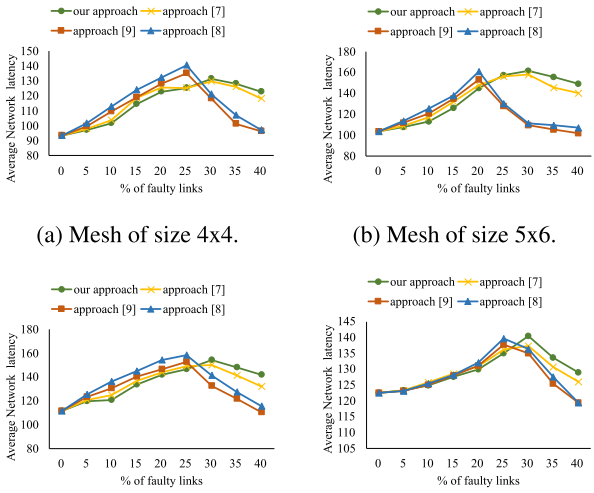
a: GROUP 1: LINK FAULTS IN MESH TOPOLOGY

In this group, simulations are performed on different sizes of mesh topology by varying the percentage of link faults present in the topology. The percentage of faulty links is calculated using the following formula:

$$Percentage\ of\ faulty\ links = \frac{No.\ of\ faulty\ links}{Total\ No.\ of\ Links}$$

Figure. 6. presents the effect of link faults on the packets delivery in different sizes of the mesh network. It is observed that as the percentage of faulty links increases in the mesh network, the percentage of packets delivered decreases. The failed links isolate some of the destinations and sources from the mesh network, which resulted in packet loss. The proposed RL-FTR algorithm delivered more packets than the algorithms proposed in [7], [8] and [9].

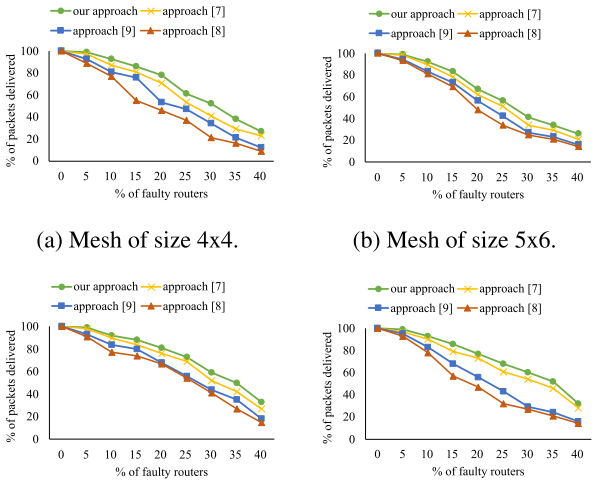
Figure. 7. presents the effect of link faults on the average network latency. It is observed that the average network latency increased till 25-30 percentage of link faults present in the mesh network because the algorithms delivered packets using fault-tolerant routing paths that are often longer compared to the fault-free routing paths. However, the average network latency for the proposed RL-FTR algorithm is less than the algorithms proposed in [7], [8] and [9]. After 30 percentage of link faults in the network, average network latency is decreased because the algorithms delivered packets only to the nearest destinations. But, it is increased in the proposed RL-FTR algorithm than the algorithms proposed



(a) Mesh of size 4x4. (b) Mesh of size 5x6.

(c) Mesh of size 6x7. (d) Mesh of size 8x8.

FIGURE 7. Change in average network latency with respect to change in the percentage of faulty links.



(a) Mesh of size 4x4. (b) Mesh of size 5x6.

(c) Mesh of size 6x7. (d) Mesh of size 8x8.

FIGURE 8. Change in percentage of packets delivered with respect to change in the percentage of faulty routers.

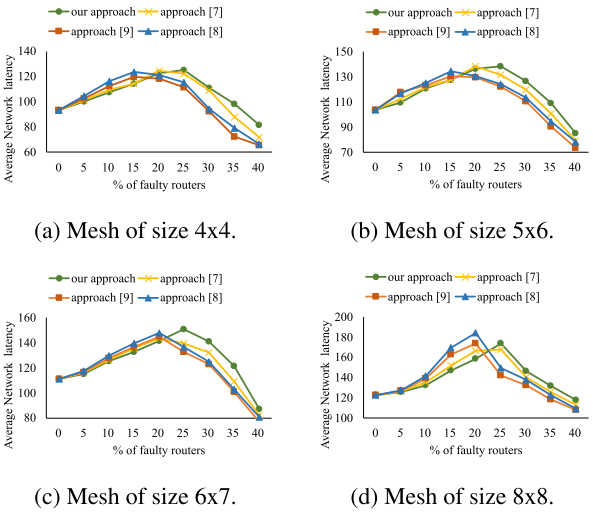
in [7], [8] and [9]. This is because the proposed RL-FTR algorithm delivered more packets to the longer distance destinations.

b: GROUP 2: ROUTER FAULTS IN MESH TOPOLOGY

Similar to the link faults, in this group, simulations are carried out by increasing the percentage of router faults in mesh topologies of different sizes. The percentage of faulty routers is calculated using the following formula:

$$Percentage\ of\ faulty\ routers = \frac{No.\ of\ faulty\ routers}{Total\ No.\ of\ routers}$$

Figure. 8 shows the variation of percentage of packets delivered with respect to the change in percentage of faulty



(a) Mesh of size 4x4. (b) Mesh of size 5x6.

(c) Mesh of size 6x7. (d) Mesh of size 8x8.

FIGURE 9. Change in average network latency with respect to change in the percentage of faulty routers.

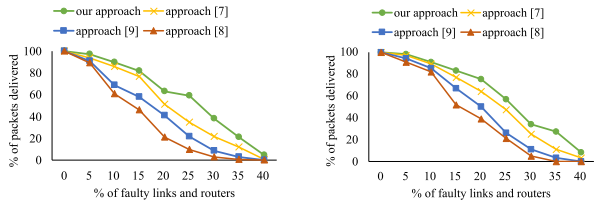
routers in mesh topologies of different sizes. Like the faulty links, the packet delivery is decreased with an increase in the percentage of faulty routers. However, the packet delivery is less for faulty routers compared to faulty links. This is because, a single faulty router can block up to four links and also it may disconnect the destination from the mesh network. In this case also, packet delivery is more for the proposed RL-FTR algorithm than the algorithms proposed in [7], [8] and [9].

Figure. 9 presents the effect of faulty routers on the average network latency in different mesh topology sizes. Similar to faulty links case, the average network latency in the proposed fault-tolerant algorithm increased till 20-25 percentage of faulty routers present in the mesh topology. The algorithms deliver packets through fault-tolerant routing paths, which are typically longer than fault-free routing paths, resulting in increased latency. After 25 percentage of faulty routers in the network, average network latency is decreased because the algorithms delivered packets only to the nearest destinations. However, in comparison to the algorithms proposed in [7], [8] and [9] , the average network latency in the proposed RL-FTR algorithm is high. This is due to efficiency of the proposed RL-FTR algorithm to deliver more packets to the longer distance destinations.

c: GROUP 3: COMBINATION OF LINK AND ROUTER FAULTS IN MESH TOPOLOGY

In this group, simulations are conducted by changing the links as well as router faults in the mesh topologies of various sizes. The percentage of faulty links and routers is calculated using the following formula:

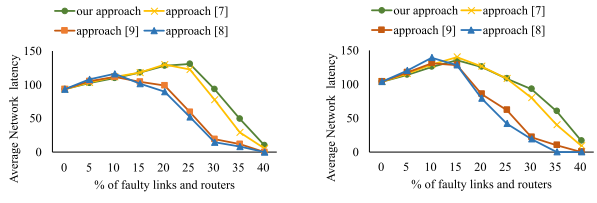
$$Percentage\ of\ faulty\ links\ and\ routers = \frac{No.\ of\ faulty\ links}{Total\ No.\ of\ Links} + \frac{No.\ of\ faulty\ routers}{Total\ No.\ of\ routers}$$



(a) Mesh of size 4x4. (b) Mesh of size 5x6.

(c) Mesh of size 6x7. (d) Mesh of size 8x8.

FIGURE 10. Change in percentage of packets delivered with respect to change in the percentage of faulty links and routers.



(a) Mesh of size 4x4. (b) Mesh of size 5x6.

(c) Mesh of size 6x7. (d) Mesh of size 8x8.

FIGURE 11. Change in average network latency with respect to change in the percentage of faulty links and routers.

The change in percentage of packets delivered and average network latency for the change in percentage of faulty links and routers are presented in Figure. 10 and Figure. 11, respectively.

From Figure. 10, it is depicted that the percentage of packets delivered are less compared to faulty links and faulty routers. Because the mesh topology has more number of faulty components (links and routers). From Figure. 11, it is evident that the average network latency is less compared with the faulty links and the faulty routers because less number of packets delivered to longer distance destinations.

In this section, the performance of the proposed RL-FTR algorithm is evaluated using NoC simulator. The percentage of packets delivered and average network latency are reported for all the experimentations.

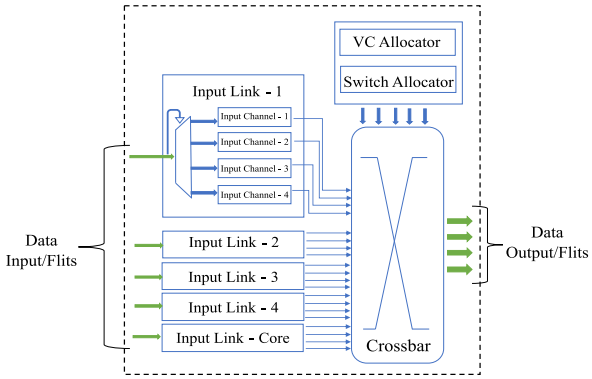


FIGURE 12. NoC router architecture overview [27].

Flit Structure			
31 (Unused)	30 (EOP)	29 (BOP)	28 -0
0	0	1	Header Flit
0	0	0	Payload Flit
0	1	0	Tailer Flit
0	1	1	Invalid Flit

FIGURE 13. Flit structure.

Header Flit									
31 (Unused)	30 (EOP)	29 (BOP)	28 (VC_ID1)	27 (VC_ID0)	26 -16 (Unused)	15-8 (Source Address)		7-0 (Destination Address)	
						15-12	11-8	7-4	3-0
0	0	1	X	X	-	Row Index	Column Index	Row Index	Column Index

FIGURE 14. Header flit structure.

B. FPGA IMPLEMENTATION

The proposed RL-FTR algorithm is implemented on FPGA to observe the real-time functioning. For FPGA implementation the router architecture is taken from [27] and routing logic is modified as per the proposed algorithm. Figure. 12 shows the overview of 5-port NoC router architecture. Out of 5 ports one port serves the core and other ports are used to connect with the neighbouring routers. Depending on the position of the router in mesh topology the number of links associated with the router varies.

The router transfers the messages in the form of flits. Flits are flow control units formed by dividing the packet into small parts. Each packet contains header, payload, and tailer flits. Header flit includes the source and destination addresses. Payload and tailer flits contain the actual message. Figure. 13 shows the structure of different flits. Each flit is 32-bit in size, and the type of the flit is defined by end-of-packet (EOP) and begin-of-packet (EOP) bits.

The structure of the header flit for mesh topology is shown in Figure. 14. It has the source and destination address fields of each 8-bits, followed by the address field, 11-bits are unused (or) reserved for future requirements. Next, 2-bits are used for virtual channel identification (VCID). For EOP and BOP, the next 2-bits are used, and the last bit is unused. We have used 8-bits to address each core. The core address

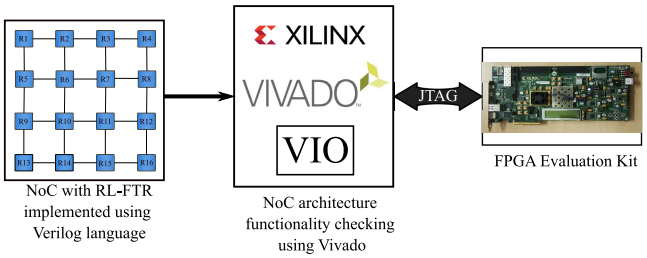


FIGURE 15. FPGA experimental setup.

field has two parts of each 4-bits, the first part of the address field holds the column index, and the second part of the address field holds the row index value. For example, 4 × 4 mesh has four rows and four columns. The index of row and column ranges from 0 to 3. So, the address of core 1 is 0000_0000 (0 × 00). Similarly, the address of core 15 is 0011_0010 (0 × 32).

1) EXPERIMENTAL SETUP

The proposed RL-FTR algorithm is developed using the Verilog programming language and integrated in place of the routing logic of the router architecture [27]. The algorithm takes the connection file consisting of mesh topology structure (without any faults or with link and routers faults) as input. It produces a Q-table with the routing path information in mesh topology based NoC. Figure. 15 shows the experimental setup of FPGA.

Mesh based NoC design of size 4 × 4 is implemented using a 5-port router. The router architecture is taken from [27]. The complete NoC design is developed using Verilog HDL programming and implemented on Xilinx Kintex-7 FPGA KC705 [28] Evaluation Kit using the Vivado tool [29]. Kintex-7 FPGA kit has a limited number of physical input/output ports. So, we have used the Vivado IP Virtual Input/Output (VIO) core to drive the inputs and to observe the output signals in real-time [30]. The output of VIO is input to design, and the input of VIO is the output of the design. Accordingly, the input core link of NoC is connected to the output of VIO to inject the packets into NoC. Similarly, the output core link of NoC is connected to the input of the VIO to observe the data received at the core link. A DIP switch present on the FPGA evaluation kit is used to manually shift the routing algorithm from fault-free to fault-tolerant and vice versa, which is shown in Figure. 16. NoC uses XY as the routing algorithm when the switch is in OFF state, and NoC uses the proposed RL-FTR algorithm for the routing when the switch is in ON state.

2) EXPERIMENTAL RESULTS ANALYSIS

The proposed RL-FTR algorithm is tested on FPGA with case studies by considering various conditions of NoC. We have used mesh topology of size 4 × 4 for the case studies, and the results are compared with the algorithms proposed in [7], [8] and [9]. Following are the case studies:

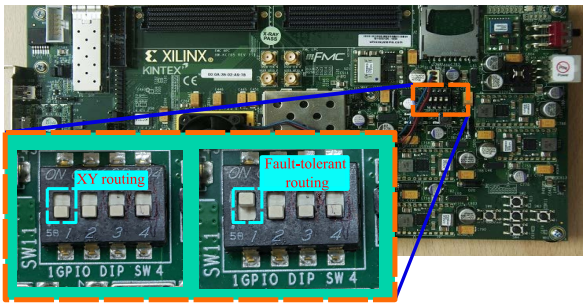


FIGURE 16. FPGA evaluation board switch configuration.

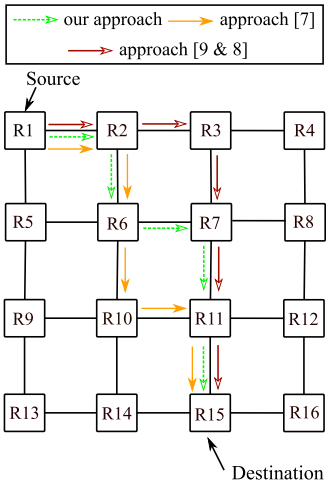
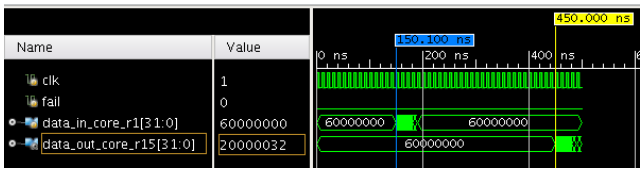
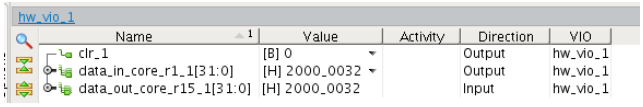


FIGURE 17. Obtained routing path between the routers R1 and R15 for mesh topology of size 4 × 4.



(a) Post-Implementation timing simulation output.

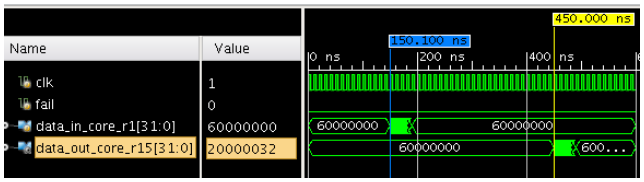


(b) VIO output

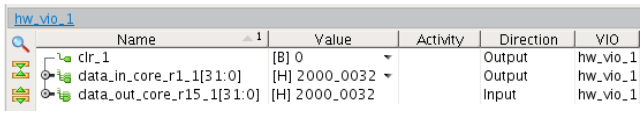
FIGURE 18. FPGA implementation of proposed RL-FTR algorithm and algorithm proposed in [7] for mesh topology of size 4 × 4 without any faults.

a: CASE1: FAULT FREE

For the case study, routers R1 and R15 are considered as the source and destination routers, respectively. Figure. 17 shows the routing path obtained from source to destination router using the proposed RL-FTR algorithm and the algorithms proposed in [7], [8] and [9]. All algorithms are taking five hops to reach the destination from the source, but the traversal paths are different.



(a) Post-Implementation timing simulation output.



(b) VIO output.

FIGURE 19. FPGA implementation of algorithms proposed in [8] and [9] for mesh topology of size 4 × 4 without any faults.

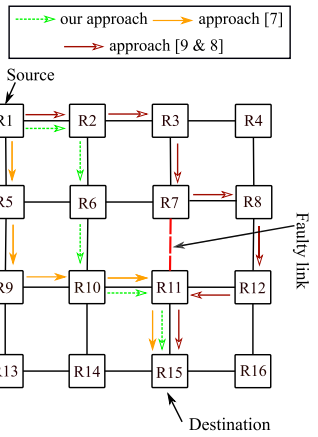


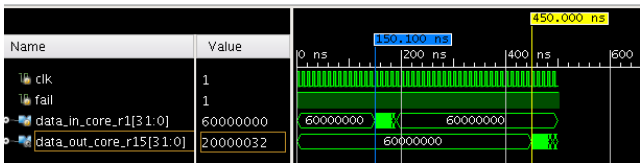
FIGURE 20. Obtained routing path between the routers R1 and R15 for mesh topology of size 4 × 4 in the presence of link fault.

Figure. 18a shows the post-implementation timing simulation output of the proposed RL-FTR algorithm and [7]. In this, the data is sent from the router R1 to the router R15, as shown in Figure. 17. The router R1 started sending the header flit at time 150.10 ns and router R15 received the header at the time 450.00 ns, i.e., it took 299.90 ns to reach router R15 from router R1. Figure. 18b shows the run-time inputs/outputs of FPGA implemented NoC design in VIO for the case taken in Figure. 17. In Figure. 18b, the input data to source router R1 is the output of VIO and the data received at the destination router R15 is observed as the input to the VIO. Similarly, Figure. 19a shows the post-implementation timing simulation output and Figure. 19b shows the run time inputs/outputs of FPGA implemented NoC design in VIO for the algorithms proposed in [8] and [9]. From Figure. 18 and 19, it is clear that all algorithms require the same amount of time to reach router R15 from router R1 in fault-free condition of mesh.

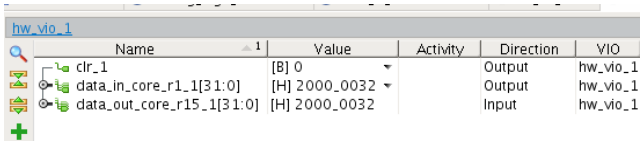
b: CASE2: LINK FAULT

In this case study, a link fault is injected in the routing path between source and destination. Here, we have considered the source-destination pair same as the fault-free case, to compare the obtained result with the fault-free case.

Figure. 20 shows the routing path obtained using the proposed RL-FTR algorithm and the algorithms proposed in [7], [8] and [9]. The routing path generated by the algorithms [8] and [9] is same. The proposed RL-FTR algorithm and [7] requires five hops to reach the destination, whereas the algorithms proposed in [8] and [9] requires seven hops to reach the destination. Compared with the algorithm in [8] and [9], the proposed RL-FTR algorithm requires two hops less, which is same as the fault-free case. Figure. 21a shows the post-implementation simulation result of the proposed RL-FTR algorithm. From Figure. 21a, it is observed that the time taken to reach R15 from R1 is equal to the time taken in fault-free case i.e., 299.90 ns. Figure. 21b and Figure. 22b show the real time inputs/outputs of FPGA implemented NoC design in VIO. In Figure. 22a, post-implementation simulation result

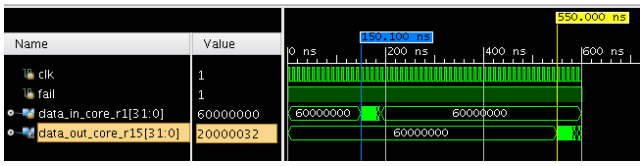


(a) Post-Implementation timing simulation output.

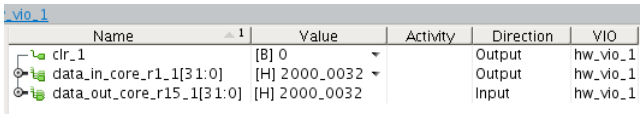


(b) VIO output

FIGURE 21. FPGA implementation of proposed RL-FTR algorithm and algorithm proposed in [7] for mesh topology of size 4 × 4 without any faults.



(a) Post-Implementation timing simulation output.



(b) VIO output.

FIGURE 22. FPGA implementation of algorithms proposed in [8] and [9] for mesh topology of size 4 × 4 in the presence of link faults.

shows the time taken to reach R15 from R1 using the algorithms proposed in [8] and [9] is 399.90 ns, which is 100ns more than that of the proposed RL-FTR algorithm and [7].

From Figure. 21a and 22a, it is evident that in the link fault case the proposed RL-FTR algorithm is providing the shortest routing path compared to the routing path provided by the algorithms proposed in [8] and [9].

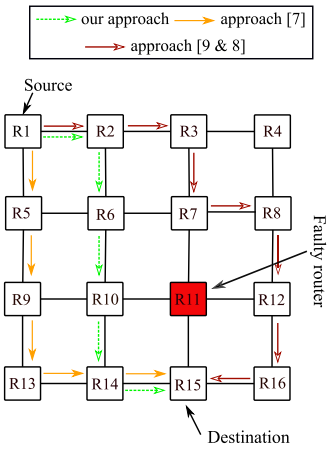
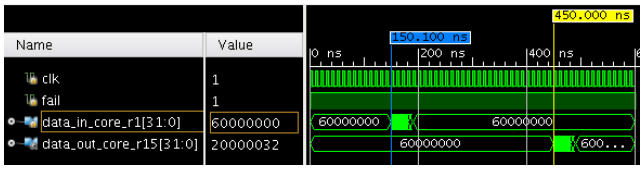
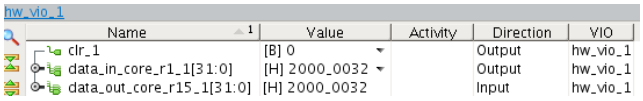


FIGURE 23. Obtained routing path between the routers R1 and R15 in mesh topology of size 4 × 4 in the presence of router fault.



(a) Post-Implementation timing simulation output.



(b) VIO output

FIGURE 24. FPGA implementation of proposed RL-FTR algorithm and algorithm proposed in [7] for mesh topology of size 4 × 4 without any faults.

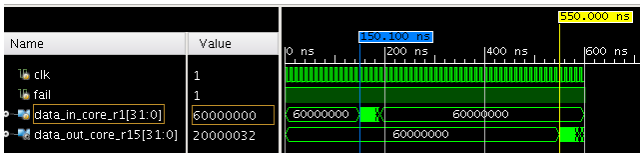
c: CASE3: ROUTER FAULT

Similar to the link fault case in this case study, a router fault is injected in the routing path between source and destination. As considered in previous cases, the same source and destination pair is used for comparative purposes. The obtained routing path is shown in Figure. 23. The routing path obtained is same for the algorithms [8] and [9]. The proposed RL-FTR algorithm requires five hops to reach the router R15 from the router R1 in router fault condition in mesh topology, which is equal to fault-free and two hops less when compared with the routing path obtained from the algorithms proposed in [8] and [9]. Figure. 24b and Figure. 25b show the input/output of the FPGA implemented NoC design in VIO.

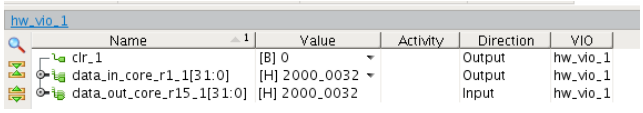
From Figure. 24a and 25a, it is observed that the proposed RL-FTR algorithm is providing the shortest routing path in case of router faults.

d: CASE4: COMBINATION OF LINK AND ROUTER FAULTS

In this case, the routing path between the source and destination routers is injected with the arbitrary router and link faults. For comparison purpose we have used the same source and destination pair, as considered in the previous cases.



(a) Post-Implementation timing simulation output.



(b) VIO output.

FIGURE 25. FPGA implementation of algorithms proposed in [8] and [9] for mesh topology of size 4 × 4 in the presence of router faults.

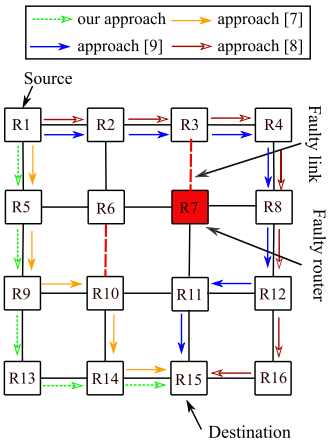


FIGURE 26. Obtained routing path between the routers R1 and R15 in mesh topology of size 4 × 4 in the presence of router fault.

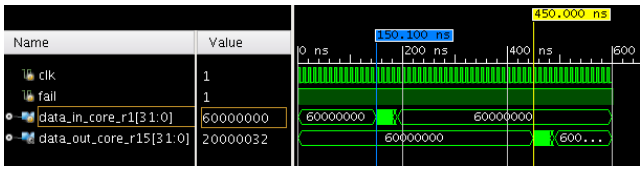
TABLE 2. Time taken by packet to travel from router R1 to router R15.

NoC Condition	RL-FTR (ns)	Algorithm proposed in [7] (ns)	Algorithm proposed in [8] (ns)	Algorithm proposed in [9] (ns)
Fault-free	299.90	299.90	299.90	299.90
Link faults	299.90	299.90	399.90	399.90
Router faults	299.90	299.90	399.90	399.90
Link and router faults	299.90	299.90	399.90	399.90

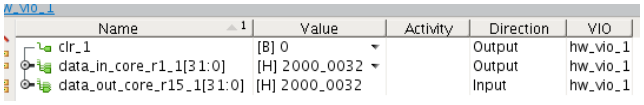
Figure. 26 depicts the obtained optimal routing path between the routers R1 and R15. In this case, the routing paths obtained from all the algorithms are different. But, the proposed RL-FTR algorithm requires 5 hops to reach the destination, which is equal to the fault-free routing path and two hops less than the path obtained using the algorithms proposed in [8] and [9]. Figure. 27b and Figure. 28b show the input/output of the FPGA implemented NoC design in VIO. From Figure. 27 and Figure. 28, it is evident that the proposed RL-FTR algorithm is providing the shortest routing path in presence of both link and routers faults. Table 2 shows the summary report of all case studies.

3) HARDWARE RESOURCE UTILIZATION

Any design implemented on FPGA consumes the available resources on FPGA. Table 3 shows the detailed hardware

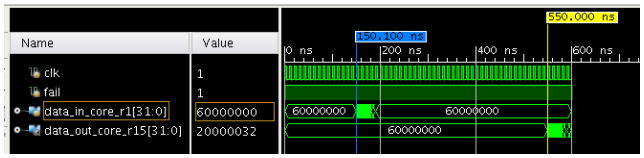


(a) Post-Implementation timing simulation output.

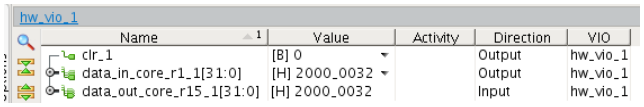


(b) VIO output

FIGURE 27. FPGA implementation of proposed RL-FTR algorithm and algorithm proposed in [7] for mesh topology of size 4 × 4 without any faults.



(a) Post-Implementation timing simulation output.



(b) VIO output.

FIGURE 28. FPGA implementation of algorithms proposed in [8] and [9] for mesh topology of size 4 × 4 in the presence of link and router faults.

TABLE 3. Hardware resource utilization report for mesh (4 × 4).

Resource	RL-FTR	Approach [7]	Approach [8]	Approach [9]
LUTs	73147	74281	72319	71137
Registers	97228	97634	97010	96610
F7 Muxes	10975	11697	10258	9297
F8 Muxes	278	284	274	248
BUFG	12	12	12	12

resource utilization report for mesh topology of size 4×4 with the proposed RL-FTR algorithm, the algorithms proposed in [7], [8] and [9]. The reports are generated from Vivado Tool.

From Table 3, it is observed that except global clock buffer (BUFG) utilization the proposed algorithm is utilizing less hardware resources than the algorithm proposed in [7] and more hardware resources than the algorithms proposed in [8] and [9]. Compared to the algorithms proposed in [8] and [9], RL-FTR algorithm has overall hardware utilization overhead is 2.27% and 8.02%, respectively. The proposed RL-FTR algorithm generates a Q-table with the information related to routing. Placing and accessing the Q-table inside the router requires more hardware resources. So, the proposed RL-FTR algorithm is using more hardware resources compared to the algorithm in [8] and [9]. But, the proposed RL-FTR is taking 2.73% less resources than the algorithm proposed in [7]. This is because in the proposed RL-FTR the every router has a individual Q-table which require less resources, whereas in [7] all the routers have a common Q-table which holds large amount of data and requires high volume of resources.

TABLE 4. On-Chip power analysis report for mesh (4 × 4).

Resource	RL-FTR (Watts)	Approach [7] (Watts)	Approach [8] (Watts)	Approach [9] (Watts)
Clocks	0.233	0.233	0.233	0.233
Signals	0.219	0.224	0.214	0.210
Logic	0.157	0.159	0.156	0.154
Total Dynamic Power	0.609	0.616	0.603	0.597
Static Power	0.163	0.163	0.163	0.163
Total Power	**0.772**	**0.779**	**0.766**	**0.760**

4) POWER ANALYSIS

A detailed power analysis for mesh topology of size 4×4 with the proposed RL-FTR algorithm, the algorithms proposed in [7], [8] and [9] is reported in Table 4.

The power analysis report is generated using Vivado tool. According to Table 3, the proposed RL-FTR algorithm is using more hardware resources than the algorithms proposed in [8] and [9], relatively the power consumption is also high for the proposed RL-FTR algorithm. Similarly, compared to the algorithm proposed in [7], RL-FTR consumes less power. Based on the power analysis reported in Table 4, it is evident that the proposed RL-FTR algorithm requires only 0.006 Watts and 0.012 Watts more power than the algorithms proposed in [8] and [9], and 0.007Watts less power than the algorithm proposed in [7]. Compared to the conventional algorithms proposed in [8] and [9], the proposed RL-FTR algorithm has very little overhead of resource utilization and power consumption. Compared to the RL based algorithm the proposed RL-FTR has less overhead of resource utilization and power consumption. However, the proposed RL-FTR algorithm is more efficient than all the compared algorithms in packet delivery and average packet latency.

In this section, functioning of the proposed RL-FTR algorithm on FPGA is observed with the case studies. The resource utilization and power analysis are reported for the FPGA implementation.

C. DISCUSSION

The proposed RL-FTR algorithm delivered an average of 4.1%, 13.1% and 21% more packets in link faults case, 4.6%, 12.5% and 17.3% more packets in router faults case, and 7.1%, 15% and 20.1% more packets in link and router faults case than the algorithms proposed in [7], [8] and [9], respectively. As the number of faults increases, the routing paths to a few nodes are blocked by faults, which results in packet loss by the proposed RL-FTR algorithm.

Overall, in both FPGA implementation and simulator, the proposed RL-FTR algorithm always provides an optimal routing path between the source and destination routers in the presence of faults. It also shows a significant improvement

over the algorithms proposed in [7], [8] and [9] against all the performance parameters.

VI. CONCLUSION

In this paper, a reinforcement learning based fault-tolerant routing algorithm is proposed and implemented on System-C based NoC simulator and FPGA. RL-FTR uses MARL with two-level Q-table. The two-level Q-table not only provides more learning information, but also mitigates outdated Q-value issue commonly experienced in large-scale systems. The experimental results show that the proposed algorithm acted well in both faulty links and routers present in the mesh topology. Performance parameters such as average network latency, packet delivery, power and hardware resource utilization are reported in this paper. In comparison with the conventional routing algorithms proposed in [8] and [9], the proposed FTR algorithm has an overall improvement of 15% and 20.1% in packets delivery, with a hardware utilization overhead of 2.27% and 8.02%, and the power consumption increased by 0.78% and 1.7%, respectively. In comparison with the RL based routing algorithm proposed in [7], the proposed FTR algorithm has an overall improvement of 7.1% in packets delivery, with 2.7% less hardware utilization, and 0.89% less power consumption. In future work, the algorithm will be extended to other topologies and implemented the same on FPGA.

REFERENCES

[1] L. Benini and G. D. Micheli, "Networks on chips: A new SoC paradigm," *Computer*, vol. 35, no. 1, pp. 70–78, Jan. 2002.
[2] V. Rantala, T. Lehtonen, and J. Plosila, *Network on Chip Routing Algorithms*. Princeton, NJ, USA: Citeseer, 2006.
[3] M. Pedram and S. Nazarian, "Thermal modeling, analysis, and management in VLSI circuits: Principles and methods," *Proc. IEEE*, vol. 94, no. 8, pp. 1487–1501, Aug. 2006.
[4] F. Tekiner, Z. Ghassemlooy, and T. Srikanth, "Comparison of the Q-routing and shortest path routing algorithms," School Eng. & Technol., Northumbria Univ., Newcastle Upon Tyne, U.K., Tech. Rep., 2004.
[5] D. Michie, D. J. Spiegelhalter, and C. Taylor, "Machine learning," *Neural Stat. Classification*, vol. 13, pp. 1–298, Feb. 1994.
[6] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.
[7] J. Samala, H. Takawale, Y. Chokhani, P. V. Bhanu, and J. Soumya, "Fault-tolerant routing algorithm for mesh based NoC using reinforcement learning," in *Proc. 24th Int. Symp. VLSI Design Test (VDAT)*, Jul. 2020, pp. 1–6.
[8] B. P. Akshay, K. M. Ganesh, D. R. Thippeswamy, V. S. Bhat, A. Vijayakumar, Y. R. Ananda, and J. Jose, "Implementation of a novel fault tolerant routing technique for mesh network on chip," in *VLSI Design Test*, S. Rajaram, N. Balamurugan, D. G. N. Rani, and V. Singh, Eds. Singapore: Springer Singapore, 2019, pp. 495–506.
[9] J. Khichar, S. Choudhary, and R. Mahar, "Fault tolerant dynamic XY–YX routing algorithm for network on-chip architecture," in *Proc. Int. Conf. Intell. Comput. Control (I2C2)*, Jun. 2017, pp. 1–6.
[10] S. Chalasani and R. V. Boppana, "Communication in multicomputers with nonconvex faults," *IEEE Trans. Comput.*, vol. 46, no. 5, pp. 616–622, May 1997.
[11] Z. Zhang, A. Greiner, and S. Taktak, "A reconfigurable routing algorithm for a fault-tolerant 2D-mesh network-on-chip," in *Proc. 45th Annu. Conf. Design Autom. (DAC)*, 2008, pp. 441–446.
[12] N. Rameshan, V. Laxmi, M. S. Gaur, M. Ahmed, and K. K. Paliwal, "Minimal path, fault tolerant, QoS aware routing with node and link failure in 2-D mesh NoC," in *Proc. IEEE 25th Int. Symp. Defect Fault Tolerance VLSI Syst.*, Oct. 2010, pp. 60–66.
[13] M. Mohtashamzadeh, L. Momeni, and A. Rezazadeh, "An innovative fault-tolerant method for 2-D mesh-based network-on-chip routing," in *Proc. UKSim 5th Eur. Symp. Comput. Modeling Simulation*, Nov. 2011, pp. 339–343.
[14] N. Chatterjee and S. Chattopadhyay, "Fault tolerant mesh based network-on-chip architecture," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2015, pp. 417–420.
[15] Y. Kurokawa and M. Fukushi, "XY based fault-tolerant routing with the passage of faulty nodes," in *Proc. 6th Int. Symp. Comput. Netw. Workshops (CANDARW)*, Nov. 2018, pp. 99–104.
[16] D. Sinha, A. Roy, K. V. Kumar, P. Kulkarni, and J. Soumya, "D<inf>n</inf>-ftr: Fault-tolerant routing algorithm for mesh based network-on-chip," in *2018 4th Int. Conf. Recent Adv. Inf. Technol. (RAIT)*, 2018, pp. 1–5.
[17] Y. Xiang, J. Meng, and D. Ma, "A q-routing based self-regulated routing scheme for network-on-chip," in *2017 IEEE 9th Int. Conf. Commun. Softw. Netw. (ICCSN)*, 2017, pp. 177–181.
[18] S.-C. Kao, C.-H.-H. Yang, P.-Y. Chen, X. Ma, and T. Krishna, "Reinforcement learning based interconnection routing for adaptive traffic optimization," in *Proc. 13th IEEE/ACM Int. Symp. Netw. Chip*, Oct. 2019, pp. 96–106, doi: 10.1145/3313231.3352369.
[19] K. Wang, A. Louri, A. Karanth, and R. Bunescu, "High-performance, energy-efficient, fault-tolerant network-on-chip design using reinforcement learning," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2019, pp. 1166–1171.
[20] H. Zheng and A. Louri, "An energy-efficient network-on-chip design using reinforcement learning," in *Proc. 56th Annu. Design Autom. Conf.*, Jun. 2019, pp. 1–6, doi: 10.1145/3316781.3317768.
[21] K. Wang, A. Louri, A. Karanth, and R. Bunescu, "IntelliNoC: A holistic design framework for energy-efficient and reliable on-chip communication for manycores," in *Proc. 46th Int. Symp. Comput. Archit.*, Jun. 2019, pp. 1–12.
[22] K. Wang and A. Louri, "CURE: A high-performance, low-power, and reliable network-on-chip design using reinforcement learning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 9, pp. 2125–2138, Sep. 2020.
[23] K. Zhang, Z. Yang, and T. Başar, "Decentralized multi-agent reinforcement learning with networked agents: Recent advances," *Frontiers Inf. Technol. Electron. Eng.*, vol. 22, no. 6, pp. 802–814, Jun. 2021.
[24] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, "An introduction to deep reinforcement learning," *Found. Trends Mach. Learn.*, vol. 11, nos. 3–4, pp. 219–354, Nov. 2018, doi: 10.1561/2200000071.
[25] W. Xia, C. Di, H. Guo, and S. Li, "Reinforcement learning based stochastic shortest path finding in wireless sensor networks," *IEEE Access*, vol. 7, pp. 157807–157817, 2019.
[26] C. J. Watkins and P. Dayan, "Technical note: Q-learning," *Mach. Learn.*, vol. 8, no. 3–4, pp. 279–292, May 1992, doi: 10.1023/A:1022676722315.
[27] S. Kundu, J. Soumya, and S. Chattopadhyay, "Design and evaluation of mesh-of-tree based network-on-chip using virtual channel router," *Microprocess. Microsyst.*, vol. 36, no. 6, pp. 471–488, Aug. 2012, doi: 10.1016/j.micpro.2012.05.012.
[28] *Xilinx Kintex-7 FPGA KC705 Evaluation Kit*. Accessed: Jan. 20, 2020. [Online]. Available: https://www.xilinx.com/products/boards-and-kits/ek-k7-kc705-g.html
[29] (2016). *Vivado Design Suite*. [Online]. Available: https://www.xilinx.com/products/design-tools/vivado.html
[30] *Virtual Input/Output (VIO)*. Accessed: Jan. 20, 2020. [Online]. Available: https://www.xilinx.com/products/intellectual-property/vio.html

SAMALA JAGADHEESH (Graduate Student Member, IEEE) received the bachelor's and master's degrees in electronics and communication engineering from Jawaharlal Nehru Technological University, Hyderabad, Telangana, India, in 2011 and 2014, respectively. He is currently pursuing the Ph.D. degree with the Department of Electrical and Electronics Engineering, Birla Institute of Technology and Science (BITS), Pilani, Hyderabad Campus, Telangana. From 2015 to 2018, he was a Faculty of the Sreenidhi Institute of Science and Technology, Hyderabad. His research interests include network-on-chip (NoC) design, fault-tolerant systems, and hardware design optimization using machine learning.

P. VEDA BHANU received the bachelor's degree in electronics and communication engineering from Jawaharlal Nehru Technological University, Hyderabad, Telangana, India, in 2015, and the master's degree in embedded systems from the National Institute of Electronics and Information Technology, Calicut, Kerala, India, in 2017. He is currently pursuing the Ph.D. degree with the Department of Electrical and Electronics Engineering, Birla Institute of Technology and Science (BITS), Pilani, Hyderabad Campus, Telangana.

From 2016 to 2017, he was an Electronic Design Intern with Panacea Medical Technologies, Bengaluru, India. From 2017 to 2020, he was a Junior Research Fellow with the Department of Electrical and Electronics Engineering, BITS Pilani, working for the Department of Science and Technology, Government of India, sponsored project. His research interests include network-on-chip (NoC) design, FPGA-based system design, optimization of performance parameters in NoC-based multiprocessor system-on-chips (MPSoCs) design, and high-performance computing.

J. SOUMYA received the bachelor's degree in electronics and communication engineering from Jawaharlal Nehru Technological University, Hyderabad, India, in 2007, and the master's and Ph.D. degrees in electronics and electrical communication engineering from the Indian Institute of Technology, Kharagpur, India, in 2010 and 2015, respectively. From 2011 to 2012, she was a Scientist "SC" at the Indian Space Research Organization (ISRO), Bengaluru, India. From 2014 to 2015, she was a Faculty Member at the National Institute of Technology (NIT), Goa, India. Since 2015, she has been an Assistant Professor with the Department of Electrical and Electronics Engineering, BITS Pilani, Hyderabad Campus, India. Her research interests include network-on-chip (NoC) design, fault-tolerant system design, NoC and processor verification, RISC-V based interconnect design, embedded systems, and real-time systems. As a Principal Investigator, she has been implementing several funded projects from DST, the Government of India, and has been collaborating with various research groups in India and abroad. Her research interests led to a credit of more than 45 publications in peer-reviewed journals and reputed international conferences held in India and abroad.

LINGA REDDY CENKERAMADDI (Senior Member, IEEE) received the master's degree in electrical engineering from the Indian Institute of Technology, New Delhi, India, in 2004, and the Ph.D. degree in electrical engineering from the Norwegian University of Science and Technology (NTNU), Trondheim, Norway, in 2011. Before joining the Ph.D. degree, he worked at Texas Instruments in mixed signal circuit design. After finishing his Ph.D. degree, he worked in radiation imaging for an atmosphere space interaction monitor (ASIM Mission to International Space Station) with the University of Bergen, Norway, from 2010 to 2012. He is currently the Group Leader of the Autonomous and Cyber-Physical Systems (ACPS) Research Group and working as a Professor with the University of Agder, Grimstad Campus, Norway. He has coauthored over 90 research publications that have been published in prestigious international journals and standard conferences. His main scientific research interests include cyber-physical systems, autonomous systems, and wireless embedded systems. He is a member of the editorial boards of various international journals, as well as the technical program committees of several IEEE conferences. He is also a principal (and co-principal) investigator of many research grants from the Norwegian Research Council.