Design of fault tolerant algorithm for network on chip router using field programmable gate array

Priti Shahane, Rakhi Kurup
Symbiosis Institute of Technology, Symbiosis International Deemed University, Pune, India

Article Info

Article history:

Received Jul 30, 2022
Revised Apr 3, 2023
Accepted May 11, 2023

Keywords:

Fault tolerant XY routing
Field programmable gate array
Network on chip
Router
System on chip

ABSTRACT

Many intellectual property (IP) modules are present in contemporary system on chips (SoCs). This could provide an issue with interconnection among different IP modules, which would limit the system's ability to scale. Traditional bus-based SoC architectures have a connectivity bottleneck, and network on chip (NoC) has evolved as an embedded switching network to address this issue. The interconnections between various cores or IP modules on a chip have a significant impact on communication and chip performance in terms of power, area latency and throughput. Also, designing a reliable fault tolerant NoC became a significant concern. In fault tolerant NoC it becomes critical to identify faulty node and dynamically reroute the packets keeping minimum latency. This study provides an insight into a domain of NoC, with intention of understanding fault tolerant approach based on the XY routing algorithm for 4×4 mesh architecture. The fault tolerant NoC design is synthesized on field programmable gate array (FPGA).

Corresponding Author:

Priti Shahane
Symbiosis Institute of Technology, Symbiosis International Deemed University
Pune Campus, Near Lupin Research Park, Lavale, Pune, Maharashtra, India
Email: pritis@sitpune.edu.in

1. INTRODUCTION

System on chip (SoC) consists of several intellectual property (IP) modules such as a general-purpose processor, input/output modules, signal processing blocks on a single chip and it is considered ideal systems with lower IP blocks as they take up lesser area and consume less power as compared to the multi-chip systems. However, the direct connections and shared bus aspect is responsible in it falling short when it comes to using it for an application with tens to thousands of IP modules. The network on chip (NoC) paradigm is advancing as an ensured answer for scalability issue of SoC due to its communication bounding restriction. With the ever developing very large-scale integration (VLSI) industry that has even crossed a billion-transistor mark, NoC circuitry is of great value.

In NoC, IPs are placed in grid (tiles) and the communication between them occurs with the help of networking protocols. This architecture has three main components: router, links, and network interface (NI). The backbone of the NoC architecture is its router. It consists of the routing algorithm which is the logic that decides the path taken by an incoming packet in order to reach its assigned destination. The NI acts as an adapter which is utilized to decouple network computations from communication and links from physical connections between routers in a network.

The router being the most critical aspect in any NoC architecture is the main focus of this study. It provides direction to organize traffic from source to destination making use of the routing algorithm. It comprises of input and output pathways in five directions north, south, east and west and core. The structure for any N×N matrix can be achieved by implementing any of the topologies such as mesh, torus, star, polygon and butterfly [1]–[3].

Another important aspect is the algorithm used in order to provide the optimum route for data packets in any given network. The routing algorithm is a crucial component of the NoCs implementation. The route between the source and destination nodes is determined by the routing algorithm. Any routing method should aim to transport packets to their destinations, distribute network traffic so as to minimize packet interference, choose a short route, reduce packet delays, and ultimately improve network performance. The routing algorithms have a major significance in NoCs as it plays crucial role on the performance, complexity and implementation cost of NoC [4], [5]. There can be broadly classified depending upon the routing decisions made into deterministic routing, in which data packets take fixed pathway towards destination based on designated standards and adaptive routing, in which the pathway depends on the network conditions.

Fault-tolerant routing is one of the key and fundamental concerns that needs to be addressed for NoCs. In NoC system, packet routing unquestionably plays a crucial role because it significantly affects system performance. It is possible that some errors may occur during system fabrication and routing of the data packets due to faulty nodes. A single faulty node disrupts data packet routing which results in the failure of the entire NoC system. Therefore, the development of fault-tolerant NoC become one of the research areas in SoC-based design [5], [6]. In this work, a fault tolerant routing algorithm is developed and applied to a 4×4 mesh based NoC router architecture to validate the algorithm.

In this paper, section 2 reviews the work done in NOC domain. Section 3 proposes single node router architecture. Faults in the NoC router are discussed in section 4. In section 5, fault tolerant routing algorithm and its implementation is discussed along with the results. Section 6 concludes with a discussion of the router's fault-tolerant algorithm implementation on the field programmable gate array (FPGA) and an emphasis on future work.

2. LITERATURE REVIEW

NoC is circuitry that has been studied keenly over the last few years by researchers due to its upper hand over SoCs in terms of better electrical properties, higher bandwidth and scalability issues. Here is a review of some of the research done that has played a significant role to this study. The basic NoC architecture explained by Bjerregaard and Mahadevan [7]. It also gives a brief description of the research areas in NoC fields. At system level, design methodology and architectural domain plays very important role for NoC design. For networks, it is the topology, protocol and flow control and at link level, reliability and packets synchronization are important.

Benini and De Micheli [8] highlighted the issues with SoC that cause the communication bottleneck. Some of these are synchronization with various network domains on the chip, physical wiring delays, shared resources, and scalability issues. It also explains why the authors believed that NoC approach is the best solution to overcome these obstacles.

Hemani *et al.* [9] explained how NoC is an interconnection of several networks computational, storage, and input/output block. This paper also discusses a honeycomb architecture implementation made up of hexagonal structures with resources at its centre and switches at its nodes. This seems to be an efficient architecture for NoC and is similar to the 2D mesh topology. Area and performance parameters of this implementation were also discussed.

Agarwal *et al.* [10] reviewed the NoC analysis for the various topologies like mesh, torus, ring, and butterfly. It gives information about the basic key aspects of NoC router architecture components like routing algorithm and its types, switching algorithm and its types, flow control methods, virtual channels, importance of quality of service (QoS). This paper also gives a brief idea about the real-world implementations of NoC. For example, Ethereal NoC by Philips.

Hesham *et al.* [11] examined the real time challenges encountered by NoC such as deadlines, performance, and cost. It also analyses the time division multiplexing (TDM) concept as a substitute to one of the commonly used circuit switching in NoC, for improving efficiency. It also compares the real time NoCs on the basis of routing, clocking and design flexibilities and analyze the parameters affecting its performance such as energy efficiency, runtime adaptability as well as scalability.

Achballah and Saoud [12] explained the NoC architecture with the help of the key characteristics topology, routing and switching. It also examines NoC dedicated simulation tools, some of which are NS-2, which is a prototyping and simulation tool. Noxim is a tool that allows user to define 2D mesh NoC architecture for throughput and latency. Orion is a simulator to estimate power and space for NoC architecture and carries out a comparison among them for modelling and hardware synthesis.

Silva *et al.* [13] talked about how NoC has overcome the constraints offered by bus based architecture of SoC with the features of parallel transactions and scalability. It also evaluates the topologies of 2D mesh and 2D torus and routing schemes such as west first, odd even (OE). The NoC based multiprocessor system-on-chip (MPSoC) shown in this paper is modelled in very high-speed integrated circuit hardware description language (VHDL) and SystemC.

Gamal *et al.* [14] implemented NoC design on Virtex 5 FPGA and analyzed on sub-module level. Soft and hard implementations were carried out and compared. It was observed that the NoCs designed for FPGA in soft implementations would utilize area better than application specific integrated circuit (ASIC). The concept of buffered and bufferless routing was explained by Cai *et al.* [15]. A comparative study of the implementation of buffered and bufferless routing processes on ASIC and FPGA were carried out. The buffered routing requires buffers to store data packets for routing. The calculation of the path that will be taken by a packet for routing is predetermined. This removes any route calculation. On the other hand, the bufferless routing was carried with the help of FLIT-BLESS routing algorithm, in this each flit is routed individually. When compared to buffered routing, the area and power taken up is more than bufferless.

Gindin *et al.* [16] proposed NoC based FPGA architecture design which is a combination of hard (implemented in silicon which includes modules like processors, multipliers, and memory interfaces) and soft (programmable units) functionalities. The approach used is a new routing scheme, weighted ordered toggled (WOT). In WOT, packets are split the same as per the XY-YX routing algorithm, with additional weights added. It is optimal when traffic requirements are not symmetric in the network.

Brugge and Khalid [17] presented an NoC router for FPGA with features such as mesh topology, XY routing algorithm, store and forward flow control, 4 ports north, south, east and west and a first in, first out (FIFO) buffer. It is a low area router design with low area overhead, which is key in NoC designs for FPGA and the router was created with flexible parameters.

Akshay *et al.* [18] presented the implementation of a new technique for mesh NoC in the realm of fault tolerance. This method makes use of the fault bits. These bits are patterned to examine the type of fault. On comparison with XY routing, the proposed method is less efficient as it takes more hops due to deflection. Another disadvantage is the increase in latency. The only advantage is the guarantee of packet delivery.

Patooghy and Miremadi [19] presented a routing algorithm for NoC with eliminating problems caused by flaws in the system. In this method, two copies of each packet are made and both are routed through separate paths- one through the standard XY routing path and the second copy through the YX routing path. This method generates limited redundant bits and traverses them through the pathway with minimum traffic. However, this XYX routing algorithm gives negligible performance and power consumption overheads. It provides almost the same reliability as that of deterministic approaches.

Pasricha *et al.* [20] presented a turn model based approach for NoC. The approach here is to combine OE and inverted OE (IOE) to get better fault tolerant results compared to existing turn model and N-random walk. The packet is sent through OE route and redundant bit through IOE route. In OE many turns are restricted. Hence, the chances of deadlock are less and the packets always take the shortest path to reach end point.

Typically, traditional fault-tolerant routing algorithms or shortest path routing algorithms base their conclusions on the predefined rules. Additionally, because to a lack of intelligence, the packets always pass via the same node on the way to the destination, causing congestion and queuing issues. Every new case necessitates human interaction in order to update the rules, which are user-defined based on commonly occurring routing problems noticed by the programmer. However, as there are more routing issues, there is a greater need to design new rules to properly address each issue, which could reduce efficiency or accuracy. And this becomes easy and more reliable with the help of machine learning (ML) technology. A recent field of research is the use of ML approaches to recommend a fault-tolerant routing algorithm for mesh-based NoC. The ML algorithm understands many routing scenarios throughout the learning process, which enable it to handle complex scenarios precisely and effectively. Samala *et al.* [21] proposed a new concept reinforcement learning (RL) based fault-tolerant routing algorithm to address the link and router faults in NoC rather than using traditional method. To develop a mesh-based NoC fault-tolerant routing system using ML becomes upcoming research domain [22], [23].

It is evident from the analysis of several NoC router architectures that buffers serve as the means to implement the majority of NoC router designs. It is always preferable the use of buffers and virtual channels to reduce network latency and contention. However major concern with the buffers is more area requirement. Therefore, there is a need to emphasize on optimized use of buffer in NoC router. This work is primarily concerned with the challenges of efficient design of the NoC router blocks in terms of fast scheduler and optimized use of buffers. As a result, the study's emphasis is on the efficient NoC router design for the area and latency optimization. Also, the evaluation of NoC router parameters targeted for implementation on FPGAs since FPGA serves as an excellent platform for hardware based NoC router implementation as it provides support for a large number of IP modules. However, due to limited logic and routing recourses, low area overhead for NoC router implementation is pivotal in FPGA based NoC. Therefore, this study deals with the efficient design of NoC router implementation on FPGA.

3. RESEARCH METHOD

In the NoC architecture, router is the most important block. It is the structure that enables the incoming packet to reach its appropriate destination in an optimal path with the help of the routing algorithm. The router architecture in study here has three main blocks: input block, scheduler block and the crossbar switch block. Each of these have a vivid role in the processing of the incoming data packet over its course of travel within the router. The novelty of the proposed design is that the design has a router with a single side buffer instead of using virtual channels in the buffers and iSLIP as a fast scheduler. Therefore, the proposed NoC router design consists of the following blocks: i) input block with single side buffer memory, ii) iSLIP scheduler, and iii) cross bar switch.

A single router consists of 5 ports architecture in which data packets can be transferred to east, west, north and south directions and the final destination is a core port of the respective node. Traditionally in routers, buffers are included in order to save the incoming packets in a queued form before they can be forwarded. This causes area and power consumption significantly affecting performance of the circuitry. In order to reduce this, single data storage buffer is implemented at every input instead. The input block is designed with a single data packet memory buffer that only stores deflected data packets in the side buffer memory. Therefore, in order to prevent data loss, it is required to have 5 buffers each for a specific direction to hold data packets in the case of contention of output ports.

The second block is the iSLIP scheduler, which determines the precise direction of data packet transfer and contains a programmable priority encoder that can vary the priority throughout each clock cycle dependent on the active node. The benefit of the iSLIP scheduler is that data packets move in a predetermined direction rather than round robin, allowing for fast scheduling. The crossbar switch in the design is responsible for physically connecting an input port to its destined output port based on the grant issued by the scheduler. And finally output port stores the data packets. The following Figure 1 shows the architecture of single node router.

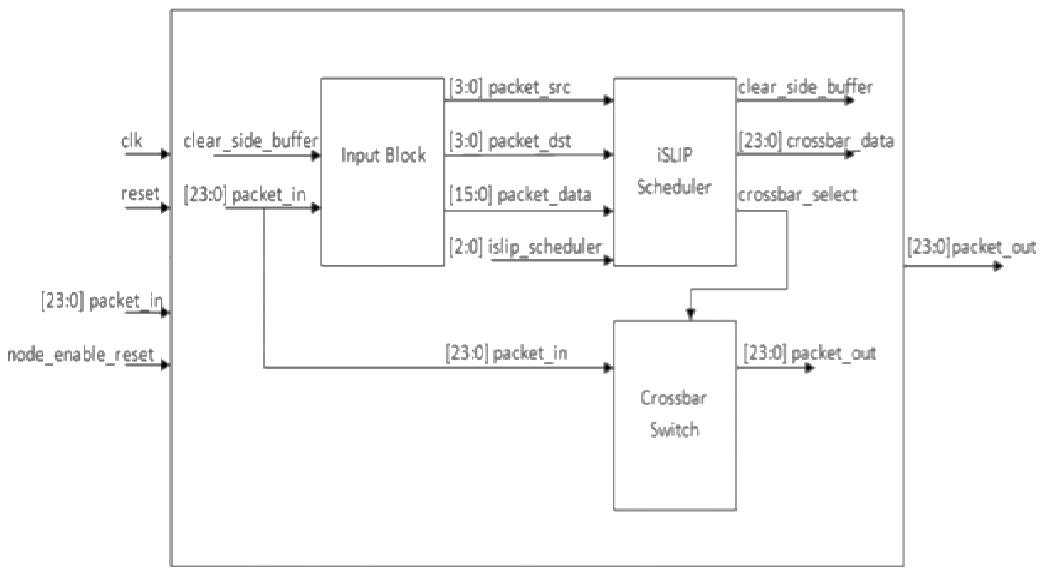


Figure 1. Architecture of single node router [24]

This single node architecture accepts a 24-bit input data packet along with clk and reset signals. The 24-bit input data can be split into 16-bits of information, along with 4-bit source address and 4-bit destination address and the output data consists of 24-bits of data received at the end of a successful cycle of data transfer. The architecture used in this study is a 4×4 mesh topology with a routing algorithm derived from XY routing as it is the most widely used routing algorithm.

4. FAULTS IN NETWORK ON CHIP

NoC architectures offer adaptability for on-chip communications through the use of fault-tolerant routing algorithms. When fault-tolerance adaptivity is added to a routing algorithm, its design complexity rises and if improperly designed, it becomes vulnerable to deadlock and other issues. The following system level faults discussed by Bengtsson *et al.* [25] in their study:

4.1. Fault due to dropping of data

These could happen in FIFO, routers, or multiplexers. For portraying the primary instance FIFO, the working has to be in details. In FIFO, the first and last areas of the FIFO are the head and tail counters. Consider the head counter gets damaged and present worth is expanded, in such a circumstance, a portion of data gets corrupted and doesn't come out. In a subsequent instance (router), assume that a fault has happened as it was steering a flit to its end point. In this circumstance, the flit gets expelled from the FIFO and does not get forwarded to any destination port. Subsequently, the packet gets dropped.

4.2. Fault due to corruption of data

It might occur in every one of segments of a switch. Consider a switch that has gotten the header flit of a data packet and has focused on a proper output port. Consequently, it sends the front bit along with its following flits, however in between this activity, an issue occurs and brings about to another data port. It can be considered that this issue a corrupt data fault as some part of the parcel is diverted away from its intended goal and will be lost because of absence of front bit which is a suitable steering data.

4.3. Fault in data travel direction

Choosing a wrong direction is the consequence of defective conduct of the network. Consider a switch with defective conduct and settles on mistaken choices instead of directing its approaching information. The flawed conduct could bring about giving unseemly select sign for router components and along these lines transferring an information to the output port that is not the intended one.

4.4. Space fault due to several copies

As examined over, flawed conduct of a switch could bring about giving improper select sign for units within the router. This type of wrong information transferring can affect the operation of several of the components in a given system. Along these lines, the information turns out from the right output port just as from at least one other output ports. In the event, if an unintended multiplexer chooses similar approaching information in light of the broken conduct, similar information will likewise go towards different yield port.

4.5. Time fault due to several copies

The different duplicates in time faults start from the FIFOs. In a FIFO with multiple copies, the fault leads to sending of old information to the ports. Such information is generally a previous parcel's flits and sending this prompts reiteration of a packet, i.e., different duplicates in time.

Along with these system level faults, routers tend to be affected by faults at architectural level in the form of link faults and node faults. Link faults occur when one or more links are unable to send the data to the next node and node faults occur when an entire node and all its surrounding links are become inactive in terms of data transfer. These faults are difficult to detect with the naked eye and can be taken care of with help of appropriate measures with the routing algorithm. These make data transfer difficult within the circuitry causing loss of information as well as power. In the study conducted, we are analysing node faults. We have generated a node fault in the 4×4 mesh architecture by disabling the node and then rerouting the data packet upon the detection of the fault.

5. FAULT TOLERANT XY ROUTING

This implemented algorithm works on the principle of firstly, detecting the fault and then rerouting the data packet to reach its intended destination via an alternate route following YX routing as soon as a fault is detected. Here, for convenience in terms of coding the node enable signal (signal from the basic building block; which acts as a grant signal for movement of data when its value is 1 and acts as a stop signal when 0) is made 0. Thereby, making a particular node as switched off/faulty. This is done so that no data transfer process is carried out through these faulty nodes and this mimics a faulty node in an NoC Router architecture. This is implemented over a 4×4 mesh topology of routers. The detailed working of fault tolerant XY routing algorithm is given as:

Step 1. Data enters in and source address is divided into 2 parts: src_reg_x, src_reg_y as well as destination is divided into 2 parts: dest_x, dest_y

Step 2. First src_y is compared to indicate whether data moves in which direction horizontally or in which direction along X-axis, i.e., east or west direction.
If src > dest, data is intended to move towards west. But it only moves in west if the node_enable signal is 1. If this signal is 0, it means that this particular node is faulty and it needs to be re-routed.
If src < dest, data is intended to move towards east. But it only moves in west if the node_enable signal is 1. If this signal is 0, it means that this particular node is faulty and it needs to be re-routed.

Step 3. For rerouting, the x-bits of source and destination are compared.

If x-bits of src < dest, only then it will be routed in north. Then it is proceeded to rerouted until it reaches a pre-destination node and then proceeds towards the original destination.
If x-bits of src > dest, only then it will be routed in south. Then it is proceeded to rerouted until it reaches a pre-destination node and then proceeds towards the original destination.

Step 4. If a fault occurs in the first leg of data traversal, i.e., during it movement along X-axis, the data packet has to be rerouted based on the y-bits. If the y-bits of source is greater, then the packet has to move in north to reach destination and thereby, fault tolerance also has to proceed in the same direction. Similarly, if the y-bits of source is smaller, then the packet has to move in south to reach destination and thereby, fault tolerance also has to proceed in the same direction.

Step 5. In the second leg of data traversal, i.e., during it movement along Y-axis, the data packet has to be rerouted based on the x-bits. Anytime a fault occurs along the Y-axis movement, it can be rerouted towards the east direction first. The only exception to this being all the nodes in the right most column.

Step 6. For rerouting, the y-bits of source and destination is compared.
If y-bits of src < dest, only then it will be routed in east. Then it is proceeded to rerouted until it reaches a pre-destination node and then proceeds towards the original destination.
If x-bits of src > dest, only then it will be routed in west. Then it is proceeded to rerouted until it reaches a pre-destination node and then proceeds towards the original destination.

As shown in the following figure, the traditional route used by the data for travelling form source (1110) to destination (0001) is to first travel in the X-direction. Once it reaches the same column as the destination, then it proceeds to move along the Y-direction until it reaches the intended destination. However, if a fault occurs at a node that is a part of the path taken to reach the destination (here fault is considered on node 1001), then upon detecting the fault, the packet reroutes its path in order to reach its destination. Data packet traversal for traditional and fault tolerant routing is given in the following Figure 2. Algorithm for routing with fault tolerant approach will work as per follows for this specific source and destination nodes with fault at node 5:

Step 1. Source: 1110 -> packet_src_x_reg = 11 and packet_src_y_reg = 10

Step 2. Destination: 0001 -> packet_dest_x = 00 and packet_dest_y = 01
packet_src_y_reg (10) > packet_dest_y (00): moves west (E -> D)
Packet_src_y_reg == packet_dest_y:

Step 3. Now, packet_src_x_reg (11) > packet_dest_x (00): moves north (D -> 9)
Data packet is supposed to continue moving in north, but node_enable of 9 = 0. Therefore, it moves west (D -> C)
packet_src_y_reg < packet_dest_y, i.e., As per XY routing it should move in East direction in order to reach the same column as destination. Due to fault at node 5, it to move in north if packet_src_y_reg == 2’b00. Therefore, it travels in north direction until it reaches 0. Data transfer is C -> 8 -> 4 -> 0.
Upon reaching 0, it moves east and reaches destination (0 -> 1).

The design is modelled in VHDL, using Xilinx ISE design suite 14.7 with Virtex 5 device xc5vlx50t-2ff665. Simulation is performed using iSIM simulator and synthesis is achieved using XST tool. The Figure 3 shows data transfer from source node 1110 to destination node 0001 with fault at node 1001. The fault tolerant algorithm reroutes the data packets from nodes 1110-1101-1100-1000-0100-0000-0001. The suggested fault tolerant algorithm provides optimized path for data packet traversal in the faulty node scenario. It may increase data latency due to rerouting of the packet.

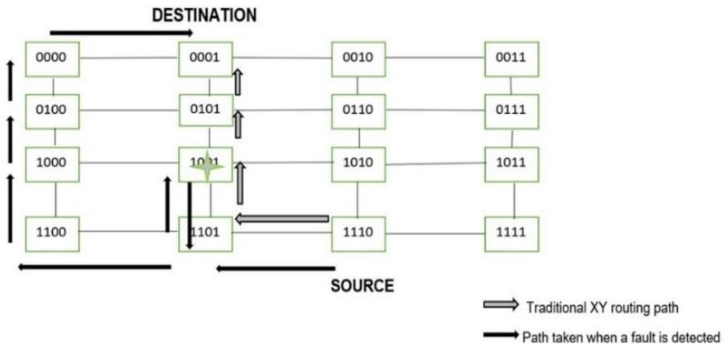
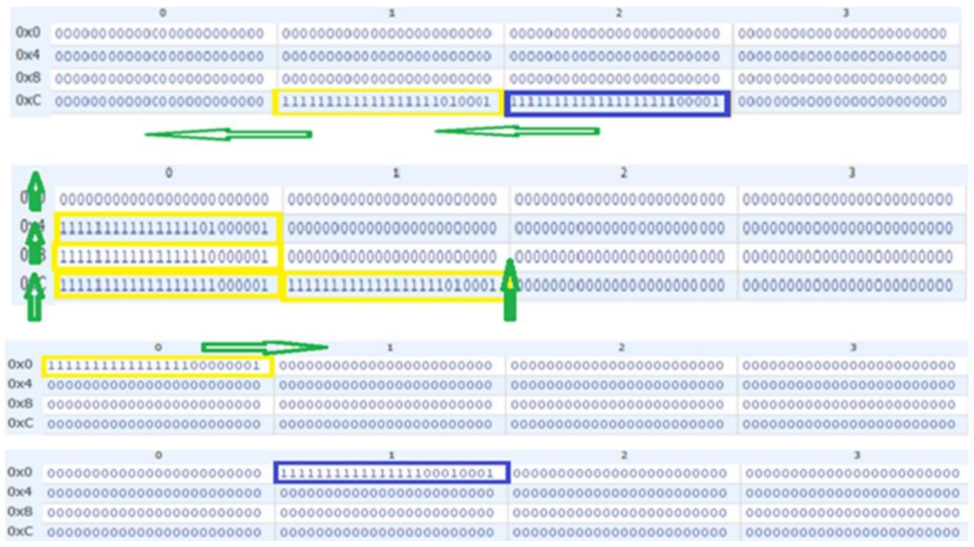


Figure 2. Fault tolerance in 4×4 mesh topology


```
                0                         1                         2                         3
0x0   000000000000000000000000  000000000000000000000000  000000000000000000000000  000000000000000000000000
0x4   000000000000000000000000  000000000000000000000000  000000000000000000000000  000000000000000000000000
0x8   000000000000000000000000  000000000000000000000000  000000000000000000000000  000000000000000000000000
0xC   000000000000000000000000  111111111111111111010001  111111111111111111100001  000000000000000000000000

                0                         1                         2                         3
0x0   000000000000000000000000  000000000000000000000000  000000000000000000000000  000000000000000000000000
0x4   111111111111111101000001  000000000000000000000000  000000000000000000000000  000000000000000000000000
0x8   111111111111111110000001  000000000000000000000000  000000000000000000000000  000000000000000000000000
0xC   111111111111111111000001  111111111111111111010001  000000000000000000000000  000000000000000000000000

                0                         1                         2                         3
0x0   111111111111111100000001  000000000000000000000000  000000000000000000000000  000000000000000000000000
0x4   000000000000000000000000  000000000000000000000000  000000000000000000000000  000000000000000000000000
0x8   000000000000000000000000  000000000000000000000000  000000000000000000000000  000000000000000000000000
0xC   000000000000000000000000  000000000000000000000000  000000000000000000000000  000000000000000000000000

                0                         1                         2                         3
0x0   000000000000000000000000  111111111111111100010001  000000000000000000000000  000000000000000000000000
0x4   000000000000000000000000  000000000000000000000000  000000000000000000000000  000000000000000000000000
0x8   000000000000000000000000  000000000000000000000000  000000000000000000000000  000000000000000000000000
0xC   000000000000000000000000  000000000000000000000000  000000000000000000000000  000000000000000000000000
```

Figure 3. Data packet transfer from source to destination

The proposed design of a 4×4 mesh topology NoC router is synthesized on Virtex 5 FPGA and the result of the synthesized design of fault-tolerant NoC algorithm is presented in Table 1. It is observed that LUT utilization is 12% and slice registers utilization is 25% on the Virtex 5 FPGA family device. The synthesized design supports a maximum operating frequency of 201 MHz. The overall power consumption is of 0.125 W using Xpower analyzer.

Table 1. Logic utilization summary

Logic utilization	Used	Available	Utilization (%)
Number of slice registers	7,276	28,800	25
Number of LUTs	3,359	28,800	12
Number of bonded IOBs	43	360	11

6. CONCLUSION

The proposed approach of fault tolerance in a 4×4 mesh topology was inspired by XY routing for mesh topology being the most commonly used routing algorithm. This methodology detects the fault when it appears in the pathway of a given source and destination. Upon detecting the fault in the path, router reroutes the data packet with new path based on the principles of XY routing. As it is based on XY routing, we can say that it takes lesser hops than other routing algorithms. This approach can be extended to even larger mesh topology-based systems. The future scope of the design is to develop fault tolerant algorithm for other NoC topology-based routers and optimize the router path to avoid data latency.

REFERENCES

[1] S. Pasricha and N. Dutt, *On-chip communication architectures: system on chip interconnect*. Elsevier, 2008. doi: 10.1016/B978-0-12-373892-9.X0001-1.

[2] É. Cota, A. de M. Amory, and M. S. Lubaszewski, *Reliability, availability and serviceability of networks-on-chip*. Boston, MA: Springer US, 2012. doi: 10.1007/978-1-4614-0791-1.

[3] J. D. Owens, W. J. Dally, R. Ho, D. N. Jayasimha, S. W. Keckler, and L.-S. Peh, "Research challenges for on-chip interconnection networks," *IEEE Micro*, vol. 27, no. 5, pp. 96–108, Sep. 2007, doi: 10.1109/MM.2007.4378787.

[4] R. Akbar, A. A. Etedalpour, and F. Safaei, "An efficient fault-tolerant routing algorithm in NoCs to tolerate permanent faults," *The Journal of Supercomputing*, vol. 72, no. 12, pp. 4629–4650, Dec. 2016, doi: 10.1007/s11227-016-1749-0.

[5] M. Fukushi and Y. Kurokawa, "A novel approach for the design of fault-tolerant routing algorithms in NoCs: passage of faulty nodes, not always detour," in *Network-on-Chip - Architecture, Optimization, and Design Explorations*, IntechOpen, 2022. doi: 10.5772/intechopen.94773.

[6] Z. Zhang, W. Serwe, J. Wu, T. Yoneda, H. Zheng, and C. Myers, "An improved fault-tolerant routing algorithm for a network-on-chip derived with formal analysis," *Science of Computer Programming*, vol. 118, pp. 24–39, Mar. 2016, doi: 10.1016/j.scico.2016.01.002.

[7] T. Bjerregaard and S. Mahadevan, "A survey of research and practices of network-on-chip," *ACM Computing Surveys*, vol. 38, no. 1, Jun. 2006, doi: 10.1145/1132952.1132953.

[8] L. Benini and G. De Micheli, "Networks on chips: a new SoC paradigm," *Computer*, vol. 35, no. 1, pp. 70–78, 2002, doi: 10.1109/2.976921.

[9] A. Hemani *et al.*, "Network on chip: qn architecture for billion transistor era," in *Proceeding of the IEEE NorChip Conference*, 2000, vol. 31, no. 20.
[10] A. Agarwal, C. Iskander, and R. Shankar, "Survey of network on chip (NoC) architectures and contributions," *Journal of Engineering, Computing and Architecture*, vol. 3, no. 1, 2009.
[11] S. Hesham, J. Rettkowski, D. Goehringer, and M. A. A. E. Ghany, "Survey on real-time networks-on-chip," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 5, pp. 1500–1517, May 2017, doi: 10.1109/TPDS.2016.2623619.
[12] A. B. Achballah and S. B. Saoud, "A survey of network on chip tools," *International Journal of Advanced Computer Science and Applications*, vol. 4, no. 9, pp. 61–67, 2013, doi: 10.14569/IJACSA.2013.040910.
[13] D. R. G. Silva, B. S. Oliveira, and F. G. Moraes, "Effects of the NoC architecture in the performance of NoC-based MPSoCs," in *2014 21st IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, Dec. 2014, pp. 431–434. doi: 10.1109/ICECS.2014.7050014.
[14] N. Gamal, H. Fahmy, Y. Ismail, T. Ismail, M. M. E. Din, and H. Mostafa, "Design guidelines for soft implementations to embedded NoCs of FPGAs," in *2016 11th International Design & Test Symposium (IDT)*, Dec. 2016, pp. 37–42. doi: 10.1109/IDT.2016.7843011.
[15] Y. Cai, K. Mai, and O. Mutlu, "Comparative evaluation of FPGA and ASIC implementations of bufferless and buffered routing algorithms for on-chip networks," in *Sixteenth International Symposium on Quality Electronic Design*, Mar. 2015, pp. 475–484. doi: 10.1109/ISQED.2015.7085472.
[16] R. Gindin, I. Cidon, and I. Keidar, "NoC-based FPGA: architecture and routing," in *First International Symposium on Networks-on-Chip (NOCS'07)*, May 2007, pp. 253–264. doi: 10.1109/NOCS.2007.31.
[17] M. Brugge and M. A. S. Khalid, "A parameterizable NoC router for FPGAs," *Journal of Computer (JCP)*, vol. 9, no. 3, pp. 519–528, 2014.
[18] B. P. Akshay *et al.*, "Implementation of a novel fault tolerant routing technique for mesh network on chip," in *International Symposium on VLSI Design and Test*, 2019, pp. 495–506. doi: 10.1007/978-981-13-5950-7_42.
[19] A. Patooghy and S. G. Miremadi, "XYX: a power & performance efficient fault-tolerant routing algorithm for network on chip," in *2009 17th Euromicro International Conference on Parallel, Distributed and Network-based Processing*, 2009, pp. 245–251. doi: 10.1109/PDP.2009.30.
[20] S. Pasricha, Y. Zou, D. Connors, and H. J. Siegel, "OE+IOE: a novel turn model based fault tolerant routing scheme for networks-on-chip," in *Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, Oct. 2010, pp. 85–94. doi: 10.1145/1878961.1878979.
[21] J. Samala, H. Takawale, Y. Chokhani, P. V. Bhanu, and J. Soumya, "Fault-tolerant routing algorithm for mesh based NoC using reinforcement learning," in *2020 24th International Symposium on VLSI Design and Test (VDAT)*, Jul. 2020, pp. 1–6. doi: 10.1109/VDAT50263.2020.9190340.
[22] 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 *Proceedings of the 13th IEEE/ACM International Symposium on Networks-on-Chip*, Oct. 2019, pp. 1–2. doi: 10.1145/3313231.3352369.
[23] K. Wang, A. Louri, A. Karanth, and R. Bunescu, "High-performance, energy-efficient, fault-tolerant network-on-chip design using reinforcement learning," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Mar. 2019, pp. 1166–1171. doi: 10.23919/DATE.2019.8714869.
[24] P. Shahane and N. Pisharoty, "Modified X–Y routing for mesh topology based NoC router on field programmable gate array," *IET Circuits, Devices & Systems*, vol. 13, no. 3, pp. 391–398, May 2019, doi: 10.1049/iet-cds.2018.5356.
[25] T. Bengtsson, S. Kumar, and Z. Peng, "Application area specific system level fault models: a case study with a simple NoC switch," in *3rd IEEE International Workshop on Electronic Design, Test and Application*, 2006, pp. 1–9.

BIOGRAPHIES OF AUTHORS

Priti Shahane is currently working as an assistant professor in in the Electronics and Telecommunication Engineering Department, Symbiosis Institute of Technology, Symbiosis International Deemed University, Pune, India. Her research is in area of VLSI design, embedded systems, and IoT. She is an IEEE senior member and fellow life member of Indian Society for technical education. She can be contacted at email: pritishahane@gmail.com

Rakhi Kurup received her M.Tech. degree from Department of Electronics and Telecommunication Engineering, Symbiosis Institute of Technology, Symbiosis International Deemed University, Pune, India. Her area of interests is in embedded systems and VLSI design. She can be contacted at email: rakhi.suresh.kurup@gmail.com.