IEEE Open Journal of
Circuits and Systems

Received 21 March 2021; revised 13 June 2021; accepted 23 June 2021. Date of current version 23 July 2021.

Digital Object Identifier 10.1109/OJCAS.2021.3095000

Self-Healing Router Approach for High-Performance Network-on-Chip

KASEM KHALIL (Member, IEEE), OMAR ELDASH (Member, IEEE), ASHOK KUMAR (Senior Member, IEEE), AND MAGDY BAYOUMI (Life Fellow, IEEE)

The Center for Advanced Computer Studies, University of Louisiana at Lafayette, Lafayette, LA 70504, USA

This article was recommended by Guest Editor L. V. Agostini.

CORRESPONDING AUTHOR: K. KHALIL (e-mail: kasem.khalil1@louisiana.edu)

ABSTRACT Network-on-Chip (NoC) is used as the communication network in many applications that use multiple cores or Processing Elements (PEs). Routers play a crucial role as connectors since a faulty router can degrade the NoC's performance and cause miscommunication between the network's components. Thus a faulty router may cause the system to fail. To avoid failure in routers in NoCs, a novel self-healing technique is proposed. Self-healing serves to recover hardware faults, and it is defined as the ability of a system to recover from its faults without any external intervention. The proposed self-healing method is to heal faulty routers and their port buffers of faults as they occur. The proposed method uses the neighboring routers of a faulty router for computation. The data packet includes three bits for routing. A neighbor's active router updates these bits according to the destination of the packet. A self-healing block is added inside each router. The proposed method also covers faulty buffers, and it uses active buffers to store the packet of a faulty one. It has been implemented and tested using VHDL and Altera Arria 10 GX FPGA. It has been found to attain improved reliability and Mean Time to Failure (MTTF) at an area overhead of 27%. It has been tested for complex NoC structure, and the results show it is practical, scalable, stable, and robust.

INDEX TERMS NoC, self-healing, hardware faults, fault tolerance, neural network, FPGA architecture.

I. INTRODUCTION

IN QUEST of increased performance, the semiconductor industry has been rapidly switching from a single microprocessor to multiple core architectures. This is enabled by a rapid reduction in the dimensions of the integrated circuit, which enables the placing of several components on a single chip. NoC is used to provide effective and efficient communication for large systems with multiple cores. NoC is used for routing between multiple cores as NoC is a flexible, scalable, and efficient interconnection technique [1], [2]. NoC can be used in many applications, such as processing components of an aircraft [3], [4] and processors in computers [5]–[8]. Apart from providing efficient communication, NoCs are expected to be power efficient [9], [10] and ideally free of fault and failures.

NoC, however, is a complex system that consists of billions of transistors and enables enormous amounts of communication, which makes it vulnerable to faults. Therefore, fault recovery is critical for increasing system reliability in such a system. The network faults can be divided into router fault, link fault, and core fault. A router fault may make the network fail at providing the needed performance. The traditional method for a fault-tolerant router uses spare or redundant routers, which increases the area overhead significantly. Sometimes, spare or redundant channels are allowed [11]. Another traditional method uses a fault-tolerant routing algorithm [12], [13]. The drawbacks of these methods are latency overhead and high area overhead that make the cost too high for mission-critical systems such as aircraft or biomedical systems. A self-healing mechanism is used to recover faults [14], [15]. Self-healing can fix faults through healing or repairing without external intervention.

Multiple processing cores can be integrated on a chip, and a faulty core can be repaired by isolating it and using a spare core instead. Such a system can be functional but with limited performance. In an NoC architecture, routers manage communications between cores [16]–[19]. In the case of a router failure, the performance degrades due to

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

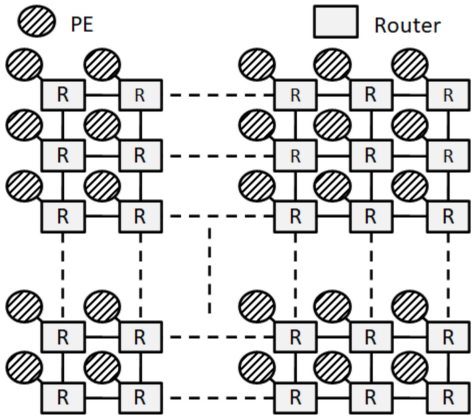


FIGURE 1. NoC architecture.

the disconnection of cores [20]. Therefore, one of the main challenges in NoC is to heal and recover faulty routers. In this paper, we focus on using self-healing to recover faulty routers at a minimal cost. The role of self-healing becomes even more critical for fault recovery in places where there may be no option for external maintenance, such as in an aircraft during flight. Self-healing performs fault repairs and improves reliability, which refers to the ability of the system to perform its function correctly and within a specific period.

NoC consists of multiple routers, and each router is connected to PE, as shown in Fig. 1. The main component of NoC for routing is the router, and the main focus of this paper is on proposing a self-healing router in NoC. The architecture of the baseline router consists of multiple components, as shown in Fig. 2. The router has five input/output ports, Virtual Channel (VC) buffers, Virtual-channel Allocation (VA), Routing Computation (RC), and Switch Allocator (SA), and Crossbar Switch (CS). The operation of each one is described as follows. The VC is the base unit of each port buffer. It is used to maximize the stored data in each port buffer. The VA component is used to make a decision for which packet request access can be the selected one. The RC block has the responsibility for routing and directing data packets towards the appropriate output channel and port. The SA component moves between VCs requesting access to the crossbar, and it gives permission to the winning packet. The central crossbar switch is a switch that makes a connection between input and output ports. It selects which input port is forwarded to which output port.

Hardware faults in NoC are divided into two classes: transient and permanent faults [21]–[23]. The transient fault is a fault that comes from external disturbance, and it may stay for a short period. The permanent fault is irretrievable physical damage in the system, and it is a continuous fault and stable with time. In this paper, we focus on permanent faults, and the reasons are described as follows. Time-Dependent Dielectric Breakdown (TDDB) is one of the sources, and it indicates insulating film breakdown due to continuous stresses to a gate oxide-film causes [24]. Negatively Biased Temperature Instability (NBTI) is another source that causes threshold voltage degradation due to a stressed transistor with negatively biased gate voltage [25]. Electromigration (EM) is also another source that occurred because of the excessive stress of current density. It causes a sudden delay increase, short, or open fault [26]. Stress Migration (SM) occurs due to excessive structural stress, and it also causes short, open, and delay faults [27]. Furthermore, Hot Carrier Injection is a source of the fault, and it causes an increase in the threshold voltage under the stress of source-drain voltage [28]. The focus of this paper to repair a faulty router.

The main focus of this work is to heal faulty routers in an NoC as a faulty router may cause isolation of its PE from the rest of the network components, and cause the degradation of performance or even failure of the system. The main contributions of this paper are to provide a self-healing method for network-on-chip based on sharing computation with neighbor routers. The proposed method addresses faults recovery for router components through the following specific contributions:

- A self-healing method,
- A packet and method for packet selection,
- Algorithm for routing in the event of a faulty router,
- A self-healing method for faulty buffers, and
- Detailed analysis and implementation results

The proposed method is scalable and incurs a small area overhead. The proposed methods are implemented on FPGA. The results show the proposed method improves mean time to failure 12.75 times more reliable than unprotected traditional baseline router and the prior works.

The remainder of this paper is organized as follows. Section II presents an overview of NoC, its application, and related work. Section III presents the proposed method of self-healing routers in NoC. Section IV discusses parameters for evaluation of self-healing. Section V presents the implementation and experimental results, followed by the conclusion in Section VI.

II. NOC BACKGROUND AND RELATED WORK

NoC is used as the enabling network for many applications such as biomedical, aerospace, and processors. For example, NoC is used for facilitating communication between the 80 cores of Teraflop processor [29]. It is used in NVIDIA Tesla V100 processor, which has 640 Tensor cores and 5000 cores that communicate using NoC [30]. NoC is also used in Tilera processor, which includes 100 cores [31]. NoC's functionality and reliability are important for many critical systems. Cores are connected to NoC via routers, and any fault that may happen in any router makes a PE isolate from the rest of PEs in the network, and the network performance decreases. The reliability of each component in an NoC can impact the overall reliability of the system [32], [33].

Motamedi *et al.* [34] present a fault-tolerant NoC technique with reconfigurable architecture. Their method is based on using redundancy to recover faulty components and increase reliability. It provides a fault recovery in the NoC's processor cores, and an application-specific configuration

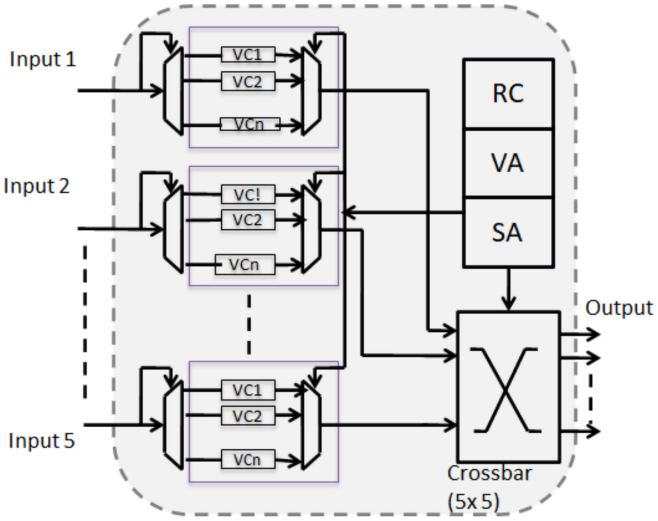


FIGURE 2. Block diagram of the router architecture.

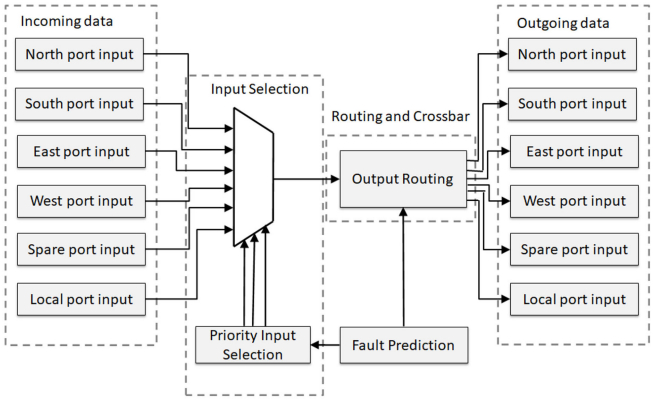


FIGURE 3. Proposed self-healing method for router.

topology is implemented for the aircraft control system. If there is a faulty component, an extra core compensates for the faulty core's operation. The method is implemented using Altera Quartus II software on a Cyclone II FPGA. The main drawback of the method is high area overhead. Heisswolf *et al.* [35] propose a method of fault-tolerance in NoC. Their method is based on a communication interface. The idea is to disable or isolate the faulty router, and another network layer is added to adapt the operations of the faulty routers. The setup of the second layer is implemented using a distributed configuration. Their method is implemented on an NoC size of 3×3 and ASIC. The method saves up to 72.2% power of the communication infrastructure power.

Fick *et al.* [36] propose a Vicis router architecture to recover permanent faults on routers and links. Their method is based on using redundancy to repair router operation where it uses bypassing a faulty path and port swapping. This technique uses a built-in self-test to reroute the data around the faulty routers and links. Vicis router has bidirectional links, and each link contains two input and output ports. In the case one port fails, the rest three ports are used for a new connection.

Wang *et al.* [37] propose a fault resilient router using a High-Performance Reliable (HPR) technique. It uses virtual channel closing for an input port, and for touting, it uses look-ahead routing. A bypass mechanism is used to tolerate the faults of the crossbar.

Constantinides *et al.* [38] propose a Bulletproof technique for repairing a faulty router. It utilizes a spatial redundancy mechanism where a backup is used for each component. The drawback of this method is that the area overhead is high. The Bulletproof provides a model of automatic clustering decomposition to achieve modularity in router architecture design. It divides the process of fault tolerance into sub-stages of detection, diagnosis, repair, and recovery.

Baloch *et al.* [39] present a defender method for a fault-tolerant router. It is based on adding other components to deal with router faults. The defender method provides fault tolerance to both the routing computation unit and the input ports by grouping the neighboring ports together.

A fault routing algorithm for network faults is presented in [12]. Their method uses an alternate bypass path that is formed around a faulty node in the network. In this way, the alternative bypass path forwards the packet to a neighboring node of a faulty node. The packet follows the XY routing algorithm through a fault-free path. The main drawbacks of this method are the complexity and scalability of wires and connections for complex systems.

Khalil *et al.* [40] propose a self-healing technique for routers in NoC. The proposed method uses a simplified spare block in each router, which allows the network to work within an expected range of performance. The spare block works as a path for the coming packets. It is used to forward the coming data to another active neighbor router. It is implemented using VHDL and ISE Xilinx Vertex 5. The method has a lower area overhead than the traditional methods that use spare routers. The method improved the reliability of the network to longer age with an area overhead of 18%.

Chatterjee and Chattopadhyay [41] present a fault-tolerant method for mesh architecture. It recovers faulty routers using a double network interface and spare links. Liu *et al.* [42] introduce a fine-grained path salvaging method for data in NoC. It splits the data path components such as links, cross-bars, and input buffer into slices instead of using redundancy. It makes a connection between multiple routers and one PE, which is implemented by designing additional ports. This is to avoid isolation of PE because of the faulty router. This method's drawbacks are that it has a large area overhead due to using spare components, links, and routers.

III. PROPOSED SELF-HEALING METHOD

The proposed self-healing method is applied for a faulty router. A faulty router isolates its local PE from the rest of PEs in the network, and the proposed method provides a

TABLE 1. Selection sequence for the output ports and the input ports.

Routing bits/Selection bits			Output port
0	0	0	Local
0	0	1	East
0	1	0	West
0	1	1	North
1	0	0	South
1	0	1	Spare

router recovery to keep the connection of the isolated PE with the others. The self-healing technique considers the faulty components of the router's components and ports' buffers. The proposed method of architecture is shown on each port in addition to the spare buffer, as shown in Fig. 3. In the case of a faulty router, the fault detection technique sends a notification to all neighbors of the faulty one. All neighbors consider this notification to update the routing bits. In NoC, each packet is 64 bits and composed of seven fields. The first two fields save the destination X-coordinate and Y-coordinate, respectively. The third and fourth fields have source X-coordinate and Y-coordinate, respectively. The fifth field has the packet sequence number. The sixth field includes the time of transmission of the packet. The last field has the payload data, which models the information inside the packet. The proposed method is based on adding three bits in the packet for routing in the case of a faulty router. These routing bits determine which port in the faulty router receives the packet from the active neighbors. The value of these bits determines the target port, "000", "0001" "010", "011", "100", "101", are used for Local, East, West, North, South, Spare ports, respectively as shown in Table 1. When the faulty router receives the packet through ports, a recovery technique is used for routing the packet in the right direction. The proposed architecture is shown in Fig. 3. It has a multiplexer with inputs from the six ports, and input selection signals are used for switching between these ports. The input selection signals are three bits that decide which port forwards its packet to the output. These values are "000", "0001" "010", "011", "100", "101" for Local, East, West, North, South, Spare ports, respectively, as shown in Table 1.

The operation is explained as follows. Each clock cycle the input selection value changes between the ports in sequence from "000" to "101", and then repeats it back. The output packet from the multiplexer is checked by the routing block to decide which output port receives this packet. It checks the last three bits as shown for each port in addition to the spare buffer, as shown in Fig. 4. The routing bits are sent to the demultiplexer, and it forwards the packet to the appropriate port according to the routing bits. The relation between the routing bits and the output ports is described in Table 1. The output port sends the packet to the neighbor router, and it uses XY routing and store-and-forward switching techniques. The algorithm of the proposed method is shown in and Algorithm 1 and Algorithm 2. For more clarification, it is assumed that the $router_9$ is faulty, as shown in Fig. 5. The

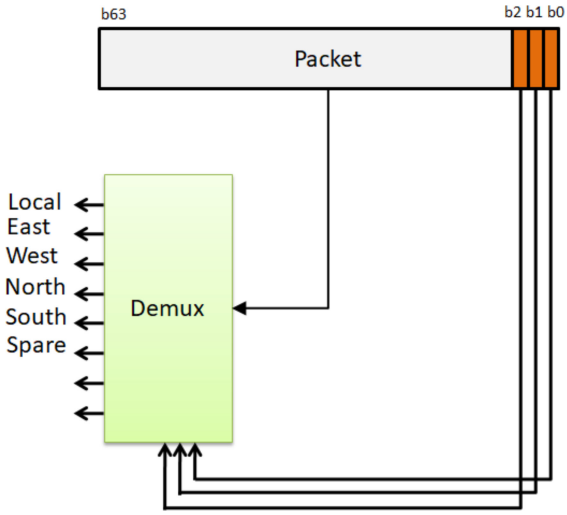


FIGURE 4. Block diagram of packet selection.

fault detection block sends a notification to the neighbor routers $router_5$, $router_8$, $router_{10}$, $router_{13}$. It is also assumed $router_{10}$ needs to send its packet to $router_9$. $Router_{10}$ checks the coordinate of the packet with the coordinate of the faulty router. If the coordinate is the same as the faulty router, the $router_{10}$ updates the routing bits to be "000". If the X-coordinate of the packet is lower than that of the faulty router, the routing bits are updated to be "010". This means the packet will forward to West port. If the Xcoordinate of the packet is the same as the faulty router, it means the packet should be sent in this column. Therefore, it checks the Y-coordinate to know which port should be used (North or South). If the Y-coordinate of the packet is higher than the faulty router, the routing bits are updated to "011" for the North port. The routing bits are updated to "100" for the South port if the Y-coordinate of the packet is lower than the faulty router. As demultiplexer is a significant component, a spare one used to avoid a situation of finding the connected demultiplexer is faulty.

For First-In-First-Out (FIFO) buffer consideration, any router has five buffers for five ports: Local, East, West, North, and South. If there is any fault in any port buffer, it leads to losing the received packet to this port. The traditional method is to use a spare buffer for each port to repair the faulty one. Therefore, five buffers are added to the router, and the area overhead of the buffer is 100% relative to the total number of buffers. The proposed self-healing method solves this challenge using a minimum area cost compared to the previous work. The proposed method uses the available buffers in addition to one spare buffer to repair any faulty buffer in the router. The proposed self-healing methods check which buffer has free slots, and then the control block sends the packet of faulty buffer to the available buffer. The proposed method reduces the area overhead by 73% compared to the traditional method, and then power consumption overhead is 2.5%. The proposed method has comparable results in delay and throughput as described in Section V.

Algorithm 1 Proposed Algorithm for Routing in Faulty Router

```
if Fault signal = '1' then
    Compare the coordinates of data packet with the current router coordinates to select output port
    if the output port = input direction port to a faulty router then
        Compare the coordinates of data packet with the coordinates of a faulty router
        if destination = faulty router address then
            routing bits = "000"
        else if destination is in the same direction of +X then
            routing bits = "001"
        else if destination is in the same direction of -X then
            routing bits = "010"
        else if destination is in the same direction of +Y then
            routing bits = "011"
        else if destination is in the same direction of -Y then
            routing bits = "100"
        end if
    end if
    Select output port to send data to next router
else
    Compare the coordinates of data packet with the current router's coordinates to select an output port
    if Destination address= faulty router address then
        Output port = Local port
    else if X_Dest − Coordinates > X_Local − Coordinates then
        Output port = East port
    else if X_Dest − Coordinates < X_Local − Coordinates then
        Output port = West port
    else if Y_Dest − Coordinates > Y_Local − Coordinates then
        Output port = North port
    else if Y_Dest − Coordinates < Y_Local − Coordinates then
        Output port = South port
    end if
end if
```

The details of the proposed method are described as follows. The method includes a FIFO controller block that has a Fault Signal (FS) input from each port. These are five signals that are coming from (*FS_E*), West port (*FS_W*), North port (*FS_N*), South port (*FS_S*), and spare (*FS_Spare*), as shown in Fig. 6. These signals come from the fault detection block. These signals are also used to indicate the availability of the router in terms of buffers storage. If the result indicates the router is full, a signal is sent to the neighbor routers to not send data to this router until receiving a signal initiated it has free slots. If the fault detection block detects a fault in any port, it raises the corresponding signal to the value of one. If there is no fault, the FS value is zero. Therefore, the FIFO controller receives a signal which indicates fault status and its location. The FIFO controller sends a request to the active ports to ask about the most available one. The FS is input to the FIFO controller, and its value is zero if there is no fault. Therefore, the FIFO controller receives this signal to indicate fault status and its location. The FIFO controller sends a request to the active ports to find the most available one. These ports send back with a grant (gnt) signal for indication of the port availability to store packets. These grant signals are for each port: East (*gnt_E*), West (*gnt_W*), North (*gnt_N*), South (*gnt_S*), Local (*gnt_L*), and Spare (*gntspare*). Each port's buffer has five input packets that come from each port in addition to the local buffer, as shown in Fig. 6. According to the available port, the controller block allows the buffer to pass the packet and save it. Then the buffer updates the number of available slots, and it sends the results back to the controller block.

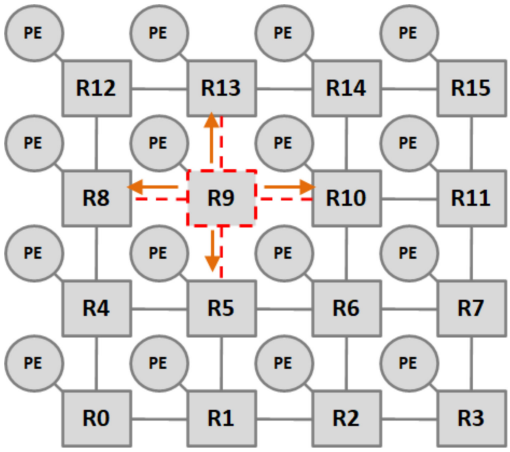


FIGURE 5. Block diagram of a faulty router in NoC.

For example, assume the east buffer is faulty. The fault detection block sets the faulty signal to one *FS_E* ='1'. The FIFO controller receives this signal, and it works to recover it by another buffer. The controller sends a request for the rest buffers (W, N, S, Spare) to check which one is available. If more than one buffers are available, the controller selects the one which has higher available slots than others based on the feedback counter from each buffer. We assume buffer North is available, it means the grant signal is one *gnt_N* ='1'. The FIFO controller sends a signal to the north buffer to receive the packet of the east port (*Pkt_E*). Then the buffer updates the number of available slots using a Free Slots Counter (FSC). This counter monitors the rest free slots in each buffer: `FSC_E`, `FSC_W`, `FSC_N`, `FSC_S`, `FSC_S`, and `FSC_Spare` for East, West, North, South, and Spare, respectively. The buffer sends the results back to the controller block. The algorithm of the proposed method is shown in Algorithm 3.

IV. PARAMETERS USED FOR EVALUATION OF SELF-HEALING

The self-healing mechanism is verified using parameters of reliability and Mean Time to Failure (MTTF). In this section, both parameters are described.

A. RELIABILITY

Reliability refers to the system's ability to perform its function successfully under some stated conditions with a given time interval. It is one of the important indexes for system performance. The definition of reliability $R(t)$ of a router in NoC is the ability of the router to perform its routing functionality within a time interval [0, t] [32, 43]. The reliability of the success of NoC is defined

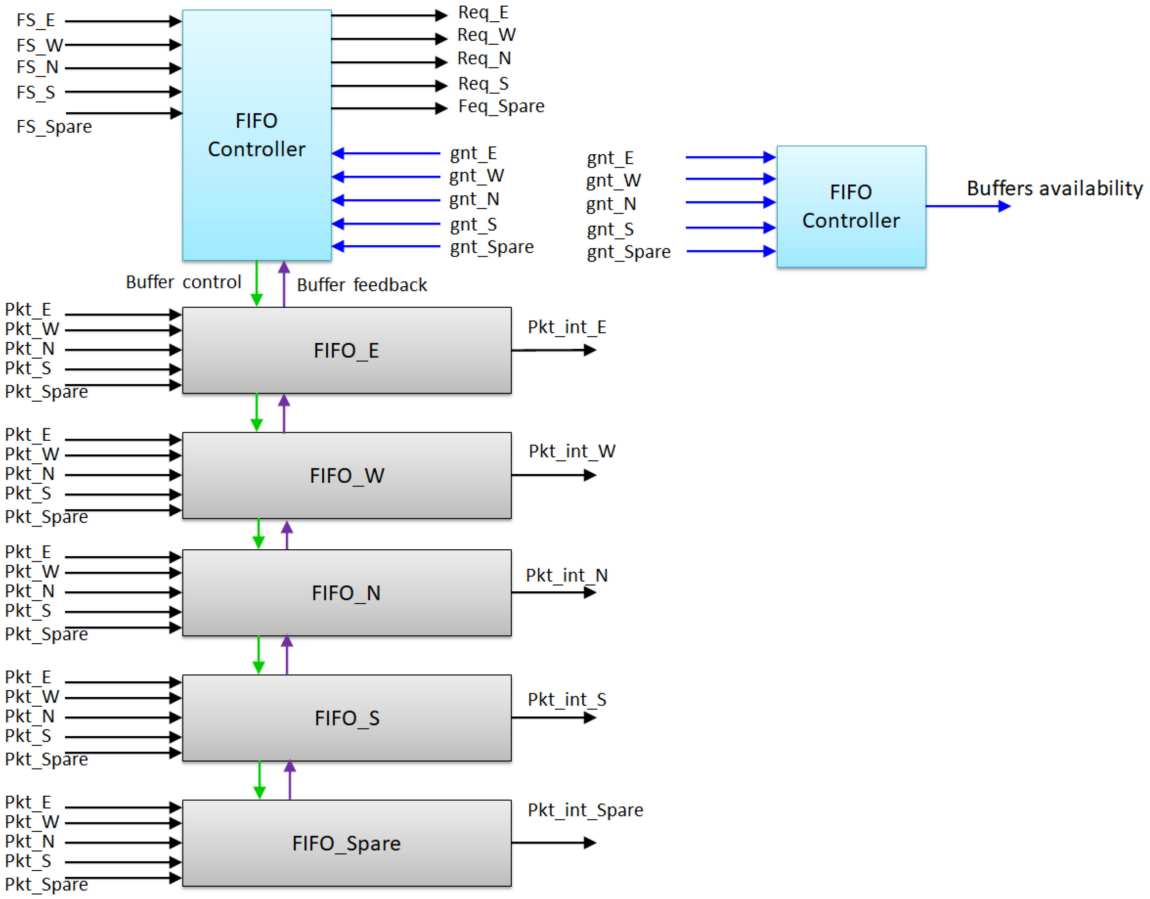


FIGURE 6. Proposed self-healing method for faulty buffers.

as shown in Equation (1).

$$p(t) = \exp^{(-\lambda t)} \tag{1}$$

where $p(t)$ is the probability of success and λ is the failure rate, and λ refers to the occurrence of faults per unit time, and it can legitimately be considered time-dependent in an electronic system. The traditional method isolates the faulty router, and the network works with the available active routers with limited performance, as shown in Fig. 7. The reliability of the NoC for the traditional $N \times N$ 2D mesh is expressed as shown in Equation (2).

$$R(t)_{\text{sys}} = \left(\exp^{(-\lambda t)}\right)^{N \times N} \tag{2}$$

where N is network dimension. The proposed method is based on dependency on the neighbor routers. Each router depends on the neighbor router, and the probability of success depends on both neighbors $(p + (1 - p)p)$ which is repeated for all routers. Therefore, the NoC reliability depends on the neighbor reliability, and it is given by Equation (3).

$$R(t)_{\text{pro sys}} = \left(\exp^{(-\lambda t)} + \left(1 - \exp^{(-\lambda t)}\right) * \exp^{(-\lambda t)}\right)^{N \times N} \tag{3}$$

A comparison between the proposed and traditional methods, is shown in Fig. 8 for (3×3) network. The reliability of the proposed method is higher than the traditional method. The proposed method also is compared with the prior methods in Section V.

B. MEAN TIME TO FAILURE (MTTF)

MTTF is the indication of the expected value of time to failure, and the NoC assessment is determined by MTTF. The MTTF, also, is the ratio of the total time of operation of all components to the number of total failures. The MTTF for the traditional method is given by Equation (4).

$$\text{MTTF} = \int_0^{\infty} \left(\exp^{(-\lambda t)}\right)^{N \times N} dt \tag{4}$$

The MTTF for the proposed method is given by Equation (5).

$$\text{MTTF} = \int_0^{\infty} \left(\exp^{(-\lambda t)} + \left(1 - \exp^{(-\lambda t)}\right) * \exp^{(-\lambda t)}\right)^{N \times N} dt. \tag{5}$$

V. IMPLEMENTATION AND EXPERIMENTAL RESULTS

The proposed self-healing method is implemented using VHDL and Altera Arria 10 GX FPGA 10AX115N2F45E1SG. The proposed method's effect on network design (size of 3×3) is studied in terms of reliability and MTTF. The reliability is calculated

Algorithm 2 Proposed Algorithm for Faulty Router

```
if Buffer fault signal ='1' then
    FIFO controller works, check algorithm. 1
end if
if the current router is faulty then
    Check which port is required to send the packet using the
    proposed approach
    if Routing bits = "000" then
        Output port = Local port
    else if Routing bits = "001" then
        if FS_E='0' then
            Output port = East port
        else
            FIFO controller selects the appropriate port
        end if
    else if Routing bits = "010" then
        if FS_W='0' then
            Output port = West port
        else
            FIFO controller selects the appropriate port
        end if
    else if Routing bits = "011" then
        if FS_N='0' then
            Output port = North port
        else
            FIFO controller selects the appropriate port
        end if
    else if Routing bits = "100" then
        if FS_S='0' then
            Output port = South port
        else
            FIFO controller selects the appropriate port
        end if
    end if
    send the packet through the selected output port
end if
```

Algorithm 3 Proposed Algorithm for Faulty Buffer

```
if FS_E= '1' then
    FIFO controller sends Req_W, Req_N, Req_S, Req_L,
    Req_Spare
    The controller receives gnt_W, gnt_N, gnt_S, gnt_L, and
    gnt_Spare
    The controller checks number of free slots in each buffer with
    grant signal
    The controller selects the freest buffer
    Buffer control signal sets the selected buffer to forward Pkt_E
    and FSC_E=FSC_E-1
else if FS_W= '1' then
    FIFO controller sends Req_E, Req_N, Req_S, Req_L,
    Req_Spare
    The controller receives gnt_E, gnt_N, gnt_S, gnt_L, and
    gnt_Spare
    The controller checks number of free slots in each buffer with
    grant signal
    The controller selects the freest buffer
    Buffer control signal sets the selected buffer to forward Pkt_W
    and FSC_W=FSC_W-1
else if FS_N= '1' then
    FIFO controller sends Req_E, Req_W, Req_S, Req_L,
    Req_Spare
    The controller receives gnt_E, gnt_W, gnt_S, gnt_L, and
    gnt_Spare
    The controller checks number of free slots in each buffer with
    grant signal
    The controller selects the freest buffer
    Buffer control signal sets the selected buffer to forward Pkt_N
    and FSC_N=FSC_N-1
else if FS_S= '1' then
    FIFO controller sends Req_E, Req_W, Req_N, Req_L,
    Req_Spare
    The controller receives gnt_E, gnt_W, gnt_N, gnt_S, and
    gnt_Spare
    The controller checks number of free slots in each buffer with
    grant signal
    The controller selects the freest buffer
    Buffer control signal sets the selected buffer to forward Pkt_S
    and FSC_S=FSC_S-1
else if FS_Spare= '1' then
    FIFO controller sends Req_E, Req_W, Req_N, Req_S,
    Req_L
    The controller receives gnt_E, gnt_W, gnt_N, gnt_S, and
    gnt_L
    The controller checks number of free slots in each buffer with
    grant signal
    The controller selects the freest buffer
    Buffer control signal sets the selected buffer to forward
    Pkt_Spare and FSC_Spare=FSC_Spare-1
end if
```

with λ = 0.315 (times/day) [44] for the network, and the proposed method's reliability is compared with the traditional method's reliability. Fault sources such as TDDB, NBTI, EM, and SM are applied to the router architecture as explained in Section I. These sources make circuits to generate signals that are expected in regular and realistic operation. A fault generator is used to allow each component to receive signals with fault effects. Some components are also can be isolated to present breakdown status. The fault generator also includes a reliability measurement unit to calculate it according to the generated failure rate and the number of routers. The experiments are carried out using a Uniform (UNI) pattern, and in this pattern, all nodes receive the same traffic distribution. The implementation of NoC on FPGA was tested using a data generator that generates data packets with a specific destination. Thus, the number of generated data packets and the number received by each router is known. Each router is used to direct the data in a desired direction. At the receiving stage, each router counts the received packets. At the end of processing, each number of received packets in each router is checked and compared with the generated number. The NOC succeeds in performing its task if the number is matched in all routers. The proposed method is compared with the baseline method. This traditional baseline method deals with fault using an isolation mechanism. It isolates the faulty router, and the network works with the available active routers with limited performance. For example, the reliability of the proposed method is improved from 0.75 to 0.98 after 10 years compared to the traditional method. The reliability is improved by 30%, which increases the age of the network. The second important parameter is MTTF. The MTTF of the proposed is improved to 96.0177 instead

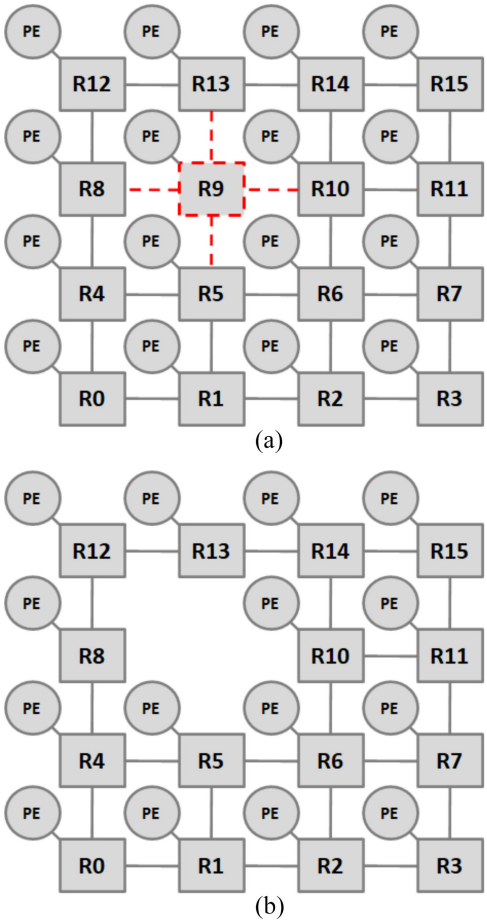


FIGURE 7. The traditional method for isolating a faulty router (a) NoC with a faulty router (b) Isolating the faulty router.

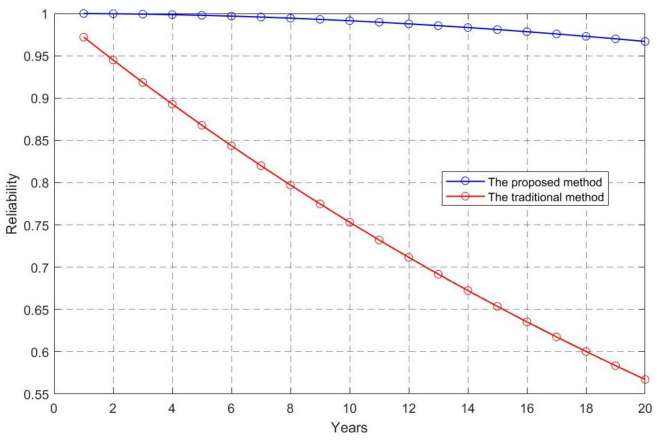


FIGURE 8. Reliability for network of size 3 × 3.

of 26.1413 for the traditional method. The MTTF is also improved using the proposed method, which is an indication of extending the time to failure. The proposed method is also tested using a Hotspot (HS) pattern where 90% of the traffic is directed to the node of the hotspot, and 10% of the traffic is distributed between all other nodes with equal distribution. In case of a normal router, the average delay

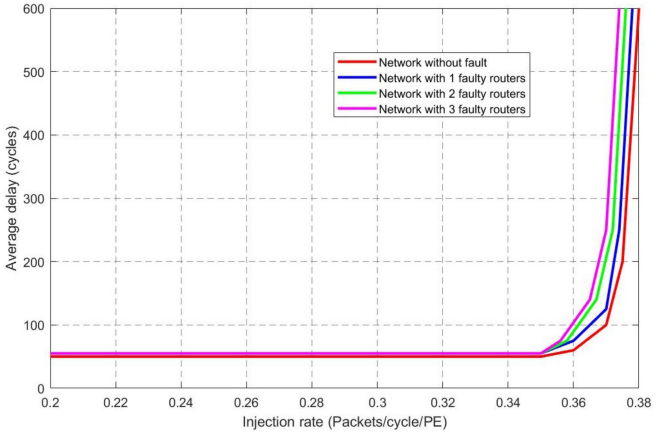


FIGURE 9. Average delay characteristics for the network using the proposed method.

and throughput are 72 cycles and 0.425, respectively, at a 0.36 injection rate. In the case the network has one faulty router, in case of a normal router, the average delay and throughput are 79 cycles and 0.41, respectively at a 0.36 injection rate. The simulation results for both the average delay and throughput have almost the same results as the UNI results.

The NoC has stable performance in terms of throughput and average delay. The throughput is the average rate of the delivered packet per unit time through a communication channel. The throughput is given by Equation (6).

$$\text{Throughput} = \frac{\sum \text{received packets}}{\text{transmission time}} \tag{6}$$

The throughput characteristics are studied where PE injects packets (typically 1000) to other PEs and also receives packets from other PEs. The Average Delay (AD) is the average time difference between the sending and receiving packets. It was measured from the sender, which generates the packets up to their reception at the destinations. It is given by Equation (7).

$$\text{AD} = \frac{\sum_0^n (\text{time of received packet} - \text{timeof sent packet})}{\text{the total number of received packets}} \tag{7}$$

Both throughput and average delay are measured to show the behavior of the network using the proposed method. These parameters are shown in Fig. 9 and Fig. 10. In the implementation, we use a uniform random synthetic traffic pattern. The injection rate is varied from 0.02 to 0.5. The simulation is repeated 20 times for each injection rate, and the average value is calculated. These results show the network maintains its stability for communication using the proposed method of healing. The average delay and throughput increase versus the injection rate. This small overhead is due to the extra computation of the controller. The proposed self-healing method is verified using a large number of nodes. It was applied on NoC from size of (3×3) to (20×20). The reliability of the traditional and the proposed self-healing method are shown in Fig. 11 and Fig. 12, respectively. The

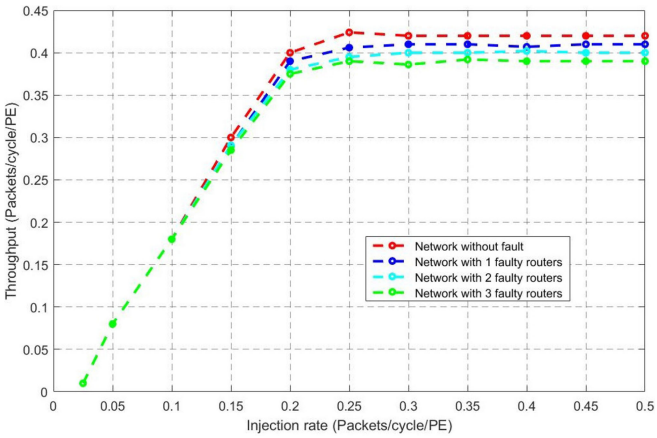


FIGURE 10. Throughput characteristics for the network using the proposed method.

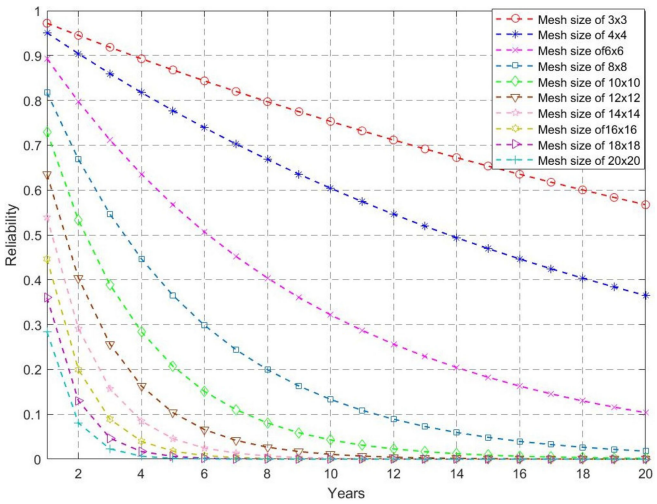


FIGURE 11. Reliability of the traditional method.

TABLE 2. Resource utilization on hardware implementation.

Parameters	The proposed method		The baseline router	
Logic Utilizing	Used	Utilization	Used	Utilization
Number of Slice LUTs	2735	9%	2370	7.8%
Number of BUFs	8	5.81%	6	4.37%
FF	541	2%	405	1.5%
Number of DSPs	14	7%	11	5.5%
Number of Block RAM	13	10%	11	7.6%
Memory LUTs	263	2%	210	1.5%

results show the stability of the proposed method for complex NoC, which can be used for many applications. A comparison between the proposed and traditional methods is shown in Fig. 13. It shows a big improvement in reliability compared to the traditional method. The results of MTTF for different network size is shown in Fig. 14. The results of reliability, MTTF, throughput, and average delay show that the proposed method is practical, scalable, and performance is stable and robust.

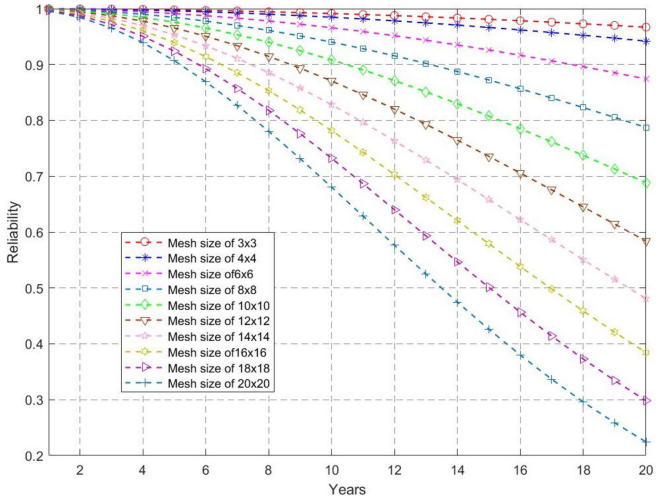


FIGURE 12. Reliability of the proposed method.

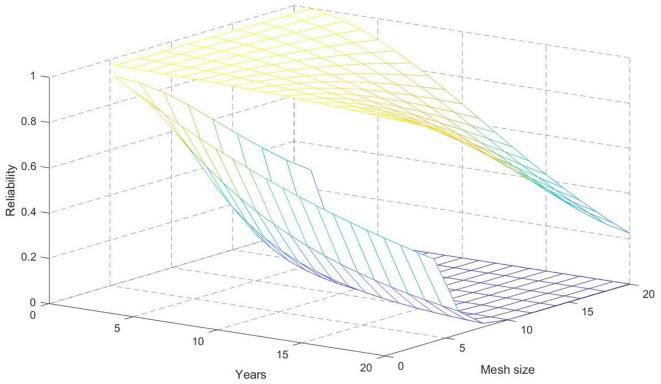


FIGURE 13. A comparison between reliability of the proposed and traditional methods.

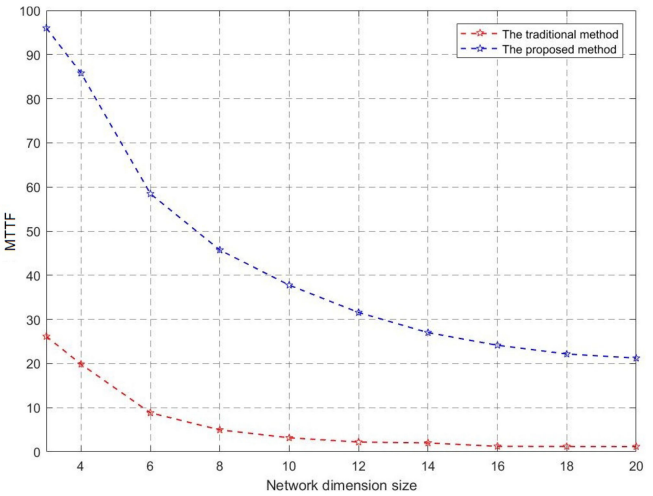


FIGURE 14. MTTF for the proposed and traditional methods.

The other important parameter for NoC is a deadlock. A deadlock is a problem where no further transportation of data packets can be occurred because of the saturation of network resources such as links or buffers. The main reason

TABLE 3. Comparison with prior works.

Ref.	Mechanism	Area Overhead	Reliability
[35]	A network layer is added to compensate for the duties of defective routers.	72.2%	N/A
[42]	A fine-grained data path salvaging technique is used by splitting data path components.	65.4%	N/A
[48]	Redundancy	50%	N/A
[36]	Vicis router architecture is used to recover permanent faults on routers and links.	42%	9.3
[37]	A fault resilient router using a High-Performance Reliable (HPR) technique.	30%	28.5
[39]	A defender method for a fault-tolerant router is used. It is based on adding other components to deal with router faults.	28%	33
[38]	A Bulletproof technique is used for repairing a faulty router. It utilizes a spatial redundancy mechanism where a backup for each component is used.	52%	3.15
The Proposed Method	**Self-healing router**	**27%**	**39.77**

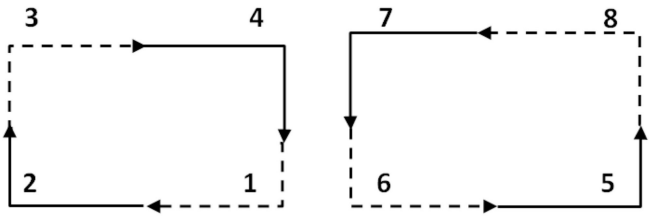


FIGURE 15. The XY Routing algorithm for possible and forbidden turns.

TABLE 4. Comparison of MTTF.

Reference	Average MTTF Gain
[37]	4.3
[39]	10.87
[49]	6
[50]	1.11
The proposed method	**12.75**

for a deadlock in NoC is the cyclic acquisition of channels in the network. It is avoided by using restrictions on where the router can store the incoming packets. These constraints are based on the incoming packet direction where the router is working using the XY routing algorithm. For example, the West buffer may contain packets directed to South, North, East, or Local. The same behavior is applied to the rest of the buffers. The routing is based on a turn model, as shown in [45]–[47], where some turns are not allowed to avoid getting complete cycles, and the network is a free deadlock. All possible turns that may happen should be considered in XY routing. For example, the turns of XY routing are West to North or South as shown in turns 5 and 4 of Fig. 15, respectively, and the turns of East to North or South, as shown in turns 2 and 7 of Fig. 15, respectively. Therefore, a complete cycle does not happen, and the proposed method is deadlock-free.

The hardware implementation of the proposed method on Altera Arria 10 GX FPGA is shown in Table 2 in terms of resource consumption. These results present the used resources which are consumed resources and utilization, which is the ratio of used resources to the total available resources. The network was implemented using Synopsys Design Compiler in 45 nm technology. The proposed method has an area overhead of 27% which is lower than the state-of-the-art methods, and it has high performance. In the proposed method, the extra block does not affect much on the power consumption because these block works when there is a fault in the network. The power consumption overhead for the proposed method is 2.5% compared to the traditional methods.

A comparison between the proposed method and previous work is shown in Table 3. The proposed method has a lower area overhead compared to [35], [42], and [48], while the reliability and MTTF are higher, as shown in Table 4. The previous work of [40] has a lower area overhead while it does not consider the faulty buffers in all ports of the router. Thus, the proposed method is more effective and efficient compared to conventional methods.

VI. CONCLUSION

This paper presented a self-healing approach for NoC. The proposed method considers the faulty routers and port's buffer in NoC. As the router is a very important unit in NoC, and the network performance may decrease due to a faulty router. The proposed method is based on using a small additional block in each router, and computation of routing bits using neighboring active routers. For the buffer, a FIFO controller block utilizes the active buffers to choose one of them to store the data packet of the faulty buffer. This selection is based on the most free buffer among the active buffers. The proposed method was implemented using VHDL and tested using Altera Arria 10 GX FPGA. The

reliability, MTTF, throughput, and average delay are studied. The results show the proposed method is very efficient and it makes the network stable. The proposed method has an area overhead of 27% with high reliability and MTTF that result in an extended network age.

REFERENCES

[1] N. L. Venkataraman, R. Kumar, and P. M. Shakeel, "Ant lion optimized bufferless routing in the design of low power application specific network on chip," *Circuits Syst. Signal Process.*, vol. 39, no. 2, pp. 961–976, 2020.

[2] M. S. Sayed, A. Shalaby, M. El-Sayed, and V. Goulart, "Flexible router architecture for network-on-chip," *Comput. Math. Appl.*, vol. 64, no. 5, pp. 1301–1310, 2012.

[3] S. Majumder, J. F. D. Nielsen, A. La Cour-Harbo, H. Schiøler, and T. Bak, "A real-time on-chip network architecture for mixed criticality aerospace systems," *Aeronaut. J.*, vol. 123, no. 1269, pp. 1788–1806, 2019.

[4] J. Malburg, K. Janson, J. Raik, and F. Dannemann, "Fault-aware performance assessment approach for embedded networks," in *Proc. IEEE 22nd Int. Symp. Design Diagn. Electron. Circuits Syst. (DDECS)*, 2019, pp. 1–4.

[5] A. Graillat, C. Maiza, M. Moy, P. Raymond, and B. D. de Dinechin, "Response time analysis of dataflow applications on a many-core processor with shared-memory and network-on-chip," in *Proc. 27th Int. Conf. Real-Time Netw. Syst.*, 2019, pp. 61–69.

[6] J. Luo, H. Zhou, Y. Zhang, N. Li, and Y. Wang, "An efficient and reliable retransmission mechanism for on-chip network of many-core processor," in *Proc. CCF Nat. Conf. Comput. Eng. Technol.*, 2019, pp. 122–135.

[7] V. Suthar, D. D. Gaitonde, A. Gupta, and J. Singh, "Placement, routing, and deadlock removal for network-on-chip using integer linear programming," U.S. Patent 10 565 346, Feb. 18, 2020.

[8] J. Harttung, E. Franz, S. Moriam, and P. Walther, "Lightweight authenticated encryption for network-on-chip communications," in *Proc. Great Lakes Symp. VLSI*, 2019, pp. 33–38.

[9] E. Wachter, L. L. Caimi, V. Fochi, D. Munhoz, and F. G. Moraes, "BrNoC: A broadcast NoC for control messages in many-core systems," *Microelectron. J.*, vol. 68, pp. 69–77, Oct. 2017.

[10] K. Khalil, O. Eldash, A. Kumar, and M. Bayoumi, "N^2 OC: Neural-network-on-chip architecture," in *Proc. 32nd IEEE Int. Syst.-Chip Conf. (SOCC)*, 2019, pp. 272–277.

[11] B. Bhowmik, J. K. Deka, S. Biswas, and B. B. Bhattacharya, "Performance-aware test scheduling for diagnosing coexistent channel faults in topology-agnostic networks-on-chip," *ACM Trans. Design Autom. Electron. Syst.*, vol. 24, no. 2, pp. 1–29, 2019.

[12] S. Priya, S. Agarwal, and H. K. Kapoor, "Fault tolerance in network on chip using bypass path establishing packets," in *Proc. 31st Int. Conf. VLSI Design 17th Int. Conf. Embedded Syst. (VLSID)*, 2018, pp. 457–458.

[13] 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)*, 2017, pp. 1–6.

[14] K. Khalil, O. Eldash, A. Kumar, and M. Bayoumi, "Intelligent fault-prediction assisted self-healing for embryonic hardware," *IEEE Trans. Biomed. Circuits Syst.*, vol. 14, no. 4, pp. 852–866, Aug. 2020.

[15] K. Khalil, O. Eldash, and M. Bayoumi, "A cost-effective self-healing approach for reliable hardware systems," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2018, pp. 1–5.

[16] D. Xiang and Q. Pan, "Low-power and high-performance adaptive routing in on-chip networks," *CCF Trans. High Perform. Comput.*, vol. 1, no. 2, pp. 92–110, 2019.

[17] P. Bogdan, T. Dumitraş, and R. Marculescu, "Stochastic communication: A new paradigm for fault-tolerant networks-on-chip," *VLSI Design*, vol. 2007, p. 17, Apr. 2007, doi: 10.1155/2007/95348.

[18] M. S. Sayed, A. Shalaby, M. E.-S. Ragab, and V. Goulart, "Congestion mitigation using flexible router architecture for network-on-chip," in *Proc. Japan-Egypt Conf. Electron. Commun. Comput.*, 2012, pp. 182–187.

[19] F. Angiolini, D. Atienza, S. Murali, L. Benini, and G. De Micheli, "Reliability support for on-chip memories using networks-on-chip," in *Proc. Int. Conf. Comput. Design*, 2006, pp. 389–396.

[20] K. Khalil, O. Eldash, and M. Bayoumi, "Self-healing router architecture for reliable network-on-chips," in *Proc. 24th IEEE Int. Conf. Electron. Circuits Syst. (ICECS)*, 2017, pp. 330–333.

[21] M. Noda, S. Kajihara, Y. Sato, K. Miyase, X. Wen, and Y. Miura, "On estimation of NBTI-induced delay degradation," in *Proc. 15th IEEE Eur. Test Symp. (ETS)*, 2010, pp. 107–111.

[22] K. Khalil, O. Eldash, A. Kumar, and M. Bayoumi, "Machine learning-based approach for hardware faults prediction," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 67, no. 11, pp. 3880–3892, Nov. 2020.

[23] T. W. Chen, K. Kim, Y. M. Kim, and S. Mitra, "Gate-oxide early life failure prediction," in *Proc. 26th IEEE VLSI Test Symp.*, 2008, pp. 111–118.

[24] T. Nigam, A. Kerber, and P. Peumans, "Accurate model for time-dependent dielectric breakdown of high-k metal gate stacks," in *Proc. IEEE Int. Rel. Phys. Symp.*, 2009, pp. 523–530.

[25] M. Ershov *et al.*, "Dynamic recovery of negative bias temperature instability in p-type metal–oxide–semiconductor field-effect transistors," *Appl. Phys. Lett.*, vol. 83, no. 8, pp. 1647–1649, 2003.

[26] G. Gielen *et al.*, "Emerging yield and reliability challenges in nanometer CMOS technologies," in *Proc. Conf. Design Autom. Test Europe*, 2008, pp. 1322–1327.

[27] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers, "The impact of technology scaling on lifetime reliability," in *Proc. DSN*, 2004, p. 177.

[28] P.-H. Chen *et al.*, "An 80 mV startup dual-mode boost converter by charge-pumped pulse generator and threshold voltage tuned oscillator with hot carrier injection," *IEEE J. Solid-State Circuits*, vol. 47, no. 11, pp. 2554–2562, Nov. 2012.

[29] S. R. Vangal *et al.*, "An 80-tile sub-100-W TeraFLOPS processor in 65-nm CMOS," *IEEE J. Solid-State Circuits*, vol. 43, no. 1, pp. 29–41, Jan. 2008.

[30] Z. Xu, X. Zhao, Z. Wang, and C. Yang, "Application-aware NOC management in GPUs multitasking," *J. Supercomput.*, vol. 75, no. 8, pp. 4710–4730, 2019.

[31] G. Passas, M. Katevenis, and D. Pnevmatikatos, "Crossbar NoCs are scalable beyond 100 nodes," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 31, no. 4, pp. 573–585, Apr. 2012.

[32] K. Khalil, O. Eldash, A. Kumar, and M. Bayoumi, "Self-healing hardware systems: A review," *Microelectron. J.*, vol. 93, Nov. 2019, Art. no. 104620.

[33] I. Schagaev, "Reliability of malfunction tolerance," in *Proc. Int. Multiconf. Comput. Sci. Inf. Technol.*, 2008, pp. 733–737.

[34] K. Motamedi, N. Ioannides, M. Rümmeli, and I. Schagaev, "Reconfigurable network on chip architecture for aerospace applications," in *Proc. 30th IFAC Workshop Real-Time Program. 4th Int. Workshop Real-Time Softw.*, 2009, pp. 131–136.

[35] J. Heisswolf *et al.*, "A novel NoC-architecture for fault tolerance and power saving," in *Proc. 29th Int. Conf. Archit. Comput. Syst.*, 2016, pp. 1–8.

[36] D. Fick, A. DeOrio, J. Hu, V. Bertacco, D. Blaauw, and D. Sylvester, "Vicis: A reliable network for unreliable silicon," in *Proc. 46th Annu. Design Autom. Conf.*, 2009, pp. 812–817.

[37] L. Wang, S. Ma, C. Li, W. Chen, and Z. Wang, "A high performance reliable NoC router," *Integration*, vol. 58, pp. 583–592, Jun. 2017.

[38] K. Constantinides *et al.*, "BulletProof: A defect-tolerant CMP switch architecture," in *Proc. 12th Int. Symp. High-Perform. Comput. Archit.*, 2006, pp. 5–16.

[39] N. K. Baloch, M. I. Baig, and M. Daneshtalab, "Defender: A low overhead and efficient fault-tolerant mechanism for reliable on-chip router," *IEEE Access*, vol. 7, pp. 142843–142854, 2019.

[40] K. Khalil, O. Eldash, A. Kumar, and M. Bayoumi, "Flexible self-healing router for reliable and high-performance network-on-chips architecture," in *Proc. 31st IEEE Int. System Chip Conf. (SOCC)*, 2018, pp. 152–157.

[41] N. Chatterjee and S. Chattopadhyay, "Fault tolerant mesh based network-on-chip architecture," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2015, pp. 417–420.

[42] C. Liu, L. Zhang, Y. Han, and X. Li, "A resilient on-chip router design through data path salvaging," in *Proc. 16th Asia South Pac. Design Autom. Conf. (ASP-DAC)*, 2011, pp. 437–442.

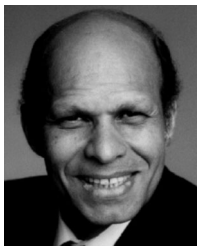
[43] Z. Zhang and Y. Wang, "Method to self-repairing reconfiguration strategy selection of embryonic cellular array on reliability analysis," in *Proc. NASA/ESA Conf. Adapt. Hardw. Syst. (AHS)*, 2014, pp. 225–232.

[44] W. J. Dally and B. P. Towles, *Principles and Practices of Interconnection Networks*. Amsterdam, The Netherlands: Elsevier, 2004.
[45] D. Xiang and W. Luo, "An efficient adaptive deadlock-free routing algorithm for torus networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 5, pp. 800–808, May 2012.
[46] M. Ebrahimi and M. Daneshtalab, "A light-weight fault-tolerant routing algorithm tolerating faulty links and routers," *Computing*, vol. 97, no. 6, pp. 631–648, 2015.
[47] C. J. Glass and L. M. Ni, "The turn model for adaptive routing," *ACM SIGARCH Comput. Archit. News*, vol. 20, no. 2, pp. 278–287, 1992.
[48] L. Xie, K. Mei, and Y. Li, "Repair: A reliable partial-redundancy-based router in NoC," in *Proc. IEEE 8th Int. Conf. Netw. Archit. Storage (NAS)*, 2013, pp. 173–177.
[49] P. Poluri and A. Louri, "Shield: A reliable network-on-chip router architecture for chip multiprocessors," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 10, pp. 3058–3070, Oct. 2016.
[50] H. J. Mohammed, W. N. Flayyih, and F. Z. Rokhani, "Tolerating permanent faults in the input port of the network on chip router," *J. Low Power Electron. Appl.*, vol. 9, no. 1, p. 11, 2019.

KASEM KHALIL (Member, IEEE) received the B.Sc. and M.Sc. degrees in electrical engineering and electronics and communications engineering from Assiut University, Asyut, Egypt, in 2009 and 2014, respectively, and the second M.Sc. degree in computer engineering and the Ph.D. degree from the Center of Advanced Computer Studies, University of Louisiana at Lafayette, USA, in 2017 and 2021, respectively. His research interests include VLSI, microelectronics, reconfigurable hardware, self-healing hardware systems, machine learning, hardware accelerators, network-on-chip, intelligent hardware systems, and the Internet of Things.

OMAR ELDASH (Member, IEEE) received the B.Sc. degree in electrical engineering from Fayoum University, Faiyum, Egypt, in 2009, and the M.Sc. degree in computer engineering from the Center of Advanced Computer Studies, University of Louisiana at Lafayette, USA, in 2016, where he is currently pursuing the Ph.D. degree. His research interests include system-on-chip, reconfigurable hardware, hardware accelerators, dynamic hardware, machine learning, and hardware accelerators.

ASHOK KUMAR (Senior Member, IEEE) received the B.Tech. degree from IIT (BHU), India, and the M.S. and Ph.D. degrees in computer engineering from the University of Southwestern Louisiana. He is currently an Associate Professor with the School of Computing and Informatics, University of Louisiana at Lafayette. His research interests include intelligent and energy-efficient software and hardware systems.



MAGDY BAYOUMI (Life Fellow, IEEE) received the B.Sc. and M.Sc. degrees in electrical engineering from Cairo University, Egypt, the M.Sc. degree in computer engineering from Washington University at St. Louis, St. Louis, and the Ph.D. degree in electrical engineering from the University of Windsor, Windsor, ON, Canada. He is currently the Department Head of the Electrical and Computer Engineering Department, University of Louisiana at Lafayette, Lafayette, LA, USA. His research interests include VLSI design and architectures, digital signal processing, and wireless ad hoc and sensor networks. He was a recipient of the 2009 IEEE Circuits and Systems Meritorious Service Award and the IEEE Circuits and Systems Society 2003 Education Award. He was the Vice President for Conferences of the IEEE Circuits and Systems Society.