Mitigating Transceiver and Token Controller Permanent Faults in Wireless Network-on-Chip

Navonil Chatterjee, Marcelo Ruaro, Kevin Martin, Jean-Philippe Diguet

Mitigating Transceiver and Token Controller Permanent Faults in Wireless Network-on-Chip

Navonil Chatterjee*, Marcelo Ruaro†, Kevin J. M. Martin† and Jean-Philippe Diguet*
*CNRS, †Université Bretagne Sud, Lab-STICC UMR 6285, Lorient, France
Email: (navonil.chatterjee, marelo.ruaro, kevin.martin, jean-philippe.diguet)@univ-ubs.fr

***Abstract*—Conventional wired Network-on-Chip (NoC) designs suffer from performance degradation due to multi-hop long-distance communication. To address such a problem, in the past decade, researchers have been focused on investigating Wireless NoC (WiNoC), which evolved as a viable solution to mitigate this communication bottleneck by using single-hop long-range wireless links. However, many researchers reported that these interconnects may suffer failure due to the complexity of implementation. Although few works in the literature tackle faults in WiNoC, none of them provides a comprehensive study related to channel access mechanisms in the presence of faults. To fill this gap, we propose a fault aware WiNoC architecture. We discuss two types of faults in wireless interconnects, namely, transceiver faults and token controller faults. We provide different fault-tolerant techniques to deal with such faults. The proposed FTWiNoC presents, on average, 17.8% and 8.9% improvement in latency compared to two different fault mitigation strategies in the literature.**

***Keywords*—** Wireless Network-on-Chip; Fault Detection; Fault Mitigating strategies.

I. Introduction

Network-on-Chip (NoC) is used as the communication backbone for multi/many-core platforms due to its scalability, flexibility, and parallelism. However, as the number of core increases wired NoC faces a serious communication bottleneck due to long multi-hop packet communication. Some works introduce long-range wires [1] and express channels [2] for on-chip long-distance communication. Both these techniques require long metal wires and need efforts to synchronize the timing of these links. Also, they require high radix routers giving rise to issues of area and power consumption. Therefore, researchers started to explore new interconnect paradigms. Researchers reported that emerging interconnect architectures can improve network performance by limiting long distance multi-hop communication observed in conventional wired NoCs [3]. There are three main emerging interconnects that are currently being explored by different research communities, which are three dimensions (3D) [4], photonic [5], and millimeter (mm) wave range wireless on-chip communication networks [6]. Among them, wireless NoC (WiNoC) has gained popularity and emerged as a viable solution that handles long-distance communications using a single-hop wireless link for large many-core systems.

One of the drawbacks of wireless interconnects is that wireless components have higher failure rates compared to other on-chip components due to their increased design complexity [7], and high utilization rate since they are used for performance enhancement [8]. Therefore, the most important issue regarding the design of a wireless interconnection framework for the large multicore system is fault tolerance, which indicates the ability of a system to function correctly even if some of its on-chip components fail. Faults in WiNoC are broadly classified into two categories, that is transient and permanent faults. Transient faults occur when one or more bits in the transmitted message become faulty. They are handled using software methods such as ECC [9]. However, it is different to deal with permanent faults, as they cause irreversible changes in the chip. It may occur during the manufacturing process or infield operation of the chip due to aging. A manufacturing test is used to detect permanent faults caused by the manufacturing process and dismiss the faulty circuits [10]. If a permanent fault emerges during the operation of the chip, it may cause degradation in system performance and may lead to system failure. Fault tolerance methods can provide means to alleviate such situations. The methods to tolerate permanent faults in wireless interface include detour and re-direction-based routing strategies [7], [8], [11], [12].

This work focuses on strategies to *detect* and *mitigate* permanent faults in WiNoC. We consider two types of faults in the WiNoC, that is, the *transceiver* faults and *token controller* faults. Failure of the transceiver section will hamper the transmission and reception of packets communicated via wireless route. We use a spare component to handle transceiver faults, where one of the transceivers is active while the other is switch-off. When the active transceiver fails, we switch it off and turn on the spare. We use token passing to ensure fair access to the wireless resource by all the network components. Therefore, the failure of the token controller may disrupt the proper functioning of the entire system. To mitigate token controller failure, we use a bypass mechanism. The packets which want to communicate via the wireless interface are detoured through the wired NoC.

The contribution of this work is twofold: (1) An online fault detection technique for WiNoC; (2) A novel method to mitigate permanent faults in WiNoC, considering both transceiver and token controller faults.

We organize the paper as follows. Section 2 describes related works. The system architecture that includes routing algorithm, wireless interface architecture, and communication protocol are discussed in section 3. The fault model and detection methods are discussed in Section 4. Fault-aware wireless NoC is presented in section 5. Section 6 presents the

results of the performance evaluation, and Section 7 concludes this work.

II. Literature Review

The paper [13] adopts complex network based architectures, along with an adaptive routing algorithm that reduces the effect of wireless link failures on the performance of the NoC. In [11], the authors present a fault-tolerant hybrid wireless NoC with a hierarchical configuration. They consider a 2D mesh architecture and divide it into clusters. Some of the nodes are selected to the cluster head (CH) used for long-range communication. In case of failure of one CH, the other CH is used for long-distance communication. A fault-tolerant table-based routing algorithm is used for short-distance communication. Also, they present a provision for long-range communication, considering all CHs have failed, using wired NoC. Similar work is done in [7], where the authors have used node-disjoint communication structures in the hybrid wireless NoC to deal with permanent faults in the wireless routers. The authors in [8] present a new WiNoC topology along with a routing mechanism to tolerate both intermittent and permanent faults on Wireless Hub (WH). The base architecture consists of a 2D mesh which is divided into subnets consisting of 16 nodes. All the nodes are connected to a single WH. In this scheme, when a WH becomes faulty, the adjacent hub is selected for the long-range packet communication. In [12], authors present link faults which include failure of wireless links. They update the routers regarding the faults and use a look-ahead detour based routing strategy to bypass the faulty links.

There are three main drawbacks to the above-reviewed works. First, the works in [7], [8], [11] redirect packets to the alternative WI, which may lead to network congestion. Second, they do not consider fault detection, which is important regarding faults in WiNoC. The works presented in [7], [8], [11], [12] do not focus on the medium access control (MAC) protocol and its effects in case of faults. Lastly, none of the works present a fault detection mechanism. In this work, we address all these issues. We use spare wireless transceivers and detour-based routing mechanisms to handle faults in WiNoC. Also, we provide fault detection, mitigation, and reconfiguration of MAC protocol for the proper functioning of the WiNoC in the presence of faults.

III. System Architecture

We consider a 2D Mesh based Wireless NoC, which is divided into equal sized non-overlapping clusters as shown in Fig 1a. Each cluster consists of 16 routers present in a 4×4 arrangement. Every cluster has a centrally placed wireless hub (WH) providing inter-cluster wireless links. It consists of the different modules required for wireless communication. Details about WH are present in Section III-B. The WH is connected to one router in the cluster, which is known as a hub connection (HC) router. Thus, all packets communicated using the WH, flow through the HC router. The architecture of our WiNoC is similar to the work [14]. We have considered a 5-port router in our system architecture, which consists of North, South, East, West, and Local port, respectively. The HC routers have a sixth port that is exclusively used to connect to the hub. This sixth port connection is used for sending/receiving packets to/from the hub for wireless transmission.

A. Routing Algorithm

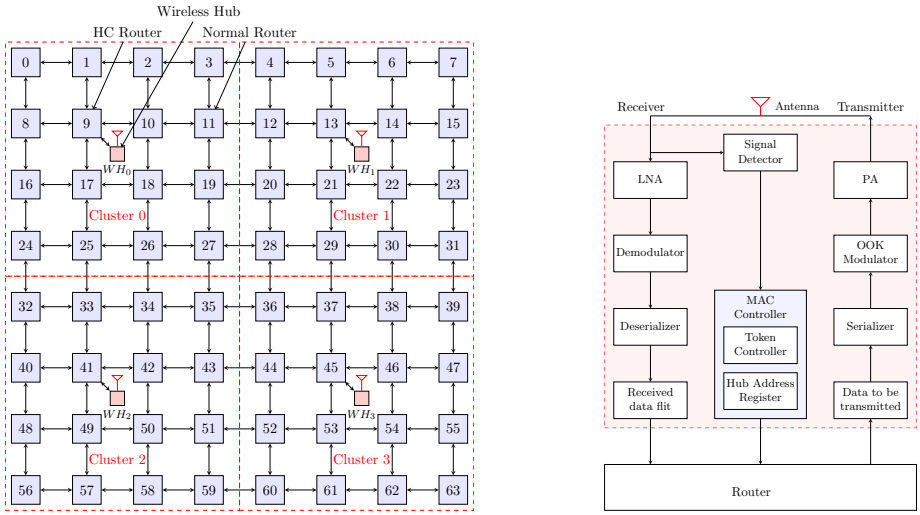
We use a threshold-based XY routing algorithm to route packets from the source to the destination node [15]. The algorithm is aware of the wireless capabilities of the NoC, including the location of the HC router. Let us briefly explain the working of this algorithm. We allocate an address to every router based on their XY coordinate. Also, every router knows their cluster id and the address of the corresponding HC router. The head flit in the packet contains the source and destination address. At the source router, the Manhattan distance between the source and destination nodes is calculated. If the Manhattan distance between the source-destination pair is greater than the threshold, the packet is routed using the WH. To route a packet through WH, first it is routed using XY routing to the HC router which sends it to the hub. The source hub transmits the packet to its destination hub using wireless transmission. Next, the packet is routed to its destination following the XY route. If the Manhattan distance is less than the threshold, the packet is routed using XY routing through the wired NoC. The threshold for wireless communication for any packet is calculated employing the equation, Eq 1.

$$TH = \alpha \times (MD(src\_r, src\_HC\_r) + MD(dst\_r, dst\_HC\_r) + WD) \quad (1)$$

Here, src_r, src_HC_r, dst_r, and dst_HC_r are source router, source HC router, destination router and destination HC router respectively. MD is the Manhattan distance, which is the number of hops between two routers. Next, WD is the wireless distance. In this work, we have considered single-hop wireless communication. Lastly, we present a new parameter α which helps to specify the desired hop count between the source and destination routers. The value of alpha is represented as $\alpha \in \mathbb{Z} | 0 < \alpha \leq ((N + M) - 1)$. N and M denoted the length and breadth of the WiNoC platform. The user can select the value of α depending upon the network traffic. In this work, we use '1' as the α value.

B. Wireless Hub Architecture and Medium Access Control

The primary components for wireless communication are an on-chip antenna and transceiver. A metal zigzag antenna is adopted from [16]. In this work, we employ a non-coherent On-Off keying (OOK) modulation scheme based transceiver for WiNoC from [17]. The transmitter (TX) circuitry consists of input buffers, a serializer, a modulator, and a power amplifier (PA). On the receiver (RX) side, a direct-conversion topology is adopted, consisting of a low noise amplifier (LNA), demodulator, deserializer, and output buffers. The input and output buffers are 32 bit wide with a buffer depth of 8 flits. As we consider the entire packet is transmitted and received during the wireless communication, these buffers are used to



(a) 8 × 8 Wireless NoC topology (b) WH Architecture

Fig. 1: The system architecture along with the WH architecture

hold the entire packet. We refer to the wireless communication architecture as wireless hub. This is shown in Fig 1b.

A Medium Access Control (MAC) mechanism is used to share wireless channels among multiple users. We use Time Division Multiple Access (TDMA) based token passing technique, where a single token is circulated among all the hubs. This scheme is implemented as follows. First, we assign a label to each of the WH. This label is called Hamiltonian labeling. The hub_label is calculated depending on the cluster coordinate and the total number of hubs present in the network. We assign two bits for representation of the hub labels as there are 4 hubs in the network. If the number of clusters increases (topology size increases), then three or more bits may be used for their representation. Therefore, the hubs belonging to clusters 0, 1, 2, 3 will be assigned as a label "00", "01", "10", and "11" respectively.

Tokens are passed among the hubs using their wireless infrastructure. First, a directed Hamiltonian path of the hub network (consisting of all the hubs in the WiNoC) is constructed. We implement a token passing scheme based on the Hamiltonian ring, where the token is passed through all the hubs only once; finally reaching the starting hub. Every hub has a register known as the hub address register (HAR), shown in Fig 1b. HAR contains the address (label) of the next hub in the ring. Once a hub receives the token, it checks whether there is any request for wireless communication. If there exists no such request, the token is passed to the next hub. Otherwise, the hub holds the token till all the flits of the packet are transferred. Each hub has the right to send one packet each time it receives the token. Once the destination hub receives the tail flit, it generates an acknowledgment (ACK) and sends it to the source hub. On receiving the ACK, the source hub gets confirmation that the packet transmission is successful and releases the token. This policy helps to avoid collisions between different hubs. Please note that a hub starts wireless transmission, if and only if the hub has the token. In case the token is not available, packets need to wait for the availability of the token.

IV. Fault Model and Detection

In this section, we discuss the different faults that we have considered in this paper, and we present the first contribution of this work, which is the fault detection in the token controller and transceiver. Note that we are assuming coarse gain faults, i.e., the failure of any component in the transmitter, receiver or token controller will lead to the failure of the hub. Also, we consider at any time instant only one hub can get faulty.

A. Token Controller Failure

In case the token controller becomes faulty, the functionality of the entire system will disrupt. The hubs require the token to get access to the wireless channel for packet transmission. After the completion of packet transmission, they transfer the token to the next hub whose address is in the HAR. In case of failure of the token controller, two situations may arise **(1)** the token controller holds the token and does not release it; **(2)** the token controller releases the token however the next hub does not receive it. In (1), one hub holds the token and all other hubs starve. While in (2), none has the token, and hence all of them starve. In both cases, the packets that need to be transferred by the WHs will wait without being served. This will cause back pressure leading to congestion in the network.

B. Token Controller Fault Detection

We detect the fault in the token controller (TC) by using three counters **PTC**= Packet Transmit Count; **THC**= Token Hold Counter; **TWC**= Token Wait Counter. We position these counters inside the dynamic fault controller (DFC) discussed in Section V-A. Every time the transmitter sends a flit, we increment the PTC. In this work, we consider every packet is 8 flits long. Therefore, PTC has 3 bits.

When a hub receives the token, THC is set. THC is incremented by 1 during the time the hub holds the token. When the hub releases the token the THC is reset. THC has 4 bits. The reason of this size is to provide extra time for receiving the ACK. The TWC is set when the hub releases the token and reset when the hub receives the token. The TWC is incremented by 1 at each clock cycle if the token is not available. The function of the TWC is to ensure every hub gets the token and no hub is starving. TWC has 8 bits. Therefore, it provides a maximum wait time (MWT) of 256 cycles. Please note that the user may increase the MWT of TWC depending on the number of WH present in the network such that it respects the fault detection procedure.

To detect whether the TC is faulty, we check THC and PTC. If PTC is 0 while THC reached the threshold, the source hub's token controller is faulty. As soon as a fault is detected in the source hub, it is switched off. Otherwise, the faulty TC could have released the token. However, the token did not reach the next hub due to fault. Therefore, THC (in the source hub) will not reset and holds the maximum value. During fault-free operation, a hub that does not have the token, its THC is reset while the TWC is set and incremented. For all other hubs, TWC is set and start counting. As the TWC for each hub starts at different times, therefore they will reach the MWT at different times. The hub that reaches the MWT first will send a token controller fault query (TCFQ) to all other WHs in the network. After receiving TCFQ, all the other hubs check the value of TWC and THC. If the value of THC and TWC do adhere to the above rule, then the hub is faulty and is switched off. All the hubs send ACK to the sender hub, except the faulty one. Using the received information, it is possible to identify the hub with a faulty TC and take the necessary measures to tolerate the fault.

C. Wireless Transceiver Failure

We consider the source transmitter (Src_Tx) or receiver (Src_Rx) faulty or destination transmitter (Dst_Tx) or receiver (Dst_Rx) faulty. Also, both Src_Tx and Src_Rx, or Dst_Tx and Dst_Rx may become faulty. In case of failure in the Src_Tx, the transfer of packets between communicating WHs will fail. The Src_Tx failure will lead to backward pressure as the packets for wireless transmission will not be served and remain waiting in the routers along the path, which ultimately will lead to network congestion. A fault in the Src_Rx will also disrupt the packet transmission process between the source and destination hub. The source hub will not receive ACK from the destination hub. The source hub will not release the token as the wireless transaction remains incomplete, leading to back pressure resulting in network congestion. Similar effect will be seen if the Dst_Rx and Dst_Tx becomes faulty.

D. Wireless Transceiver Fault Detection

We use the registers PTC and THC for the detection of faults in transceiver. Source Hub acquires the token and checks if there is any packet in the input buffer. If the input buffer is empty, the token is released else packet transmission begins. We set the PTC and THC. Source hub sends the flits in the packet one by one to the destination hub. We increment THC and PTC. Next, we check PTC. If PTC is equal to the packet's size, packet transmission terminates. On receiving the tail flit in the destination hub, it sends an ACK to the source hub. The source hub, on receiving the ACK, releases the token as packet transmission is successful. Thereafter, we reset PTC and THC. However, if ACK is not received, we check THC. If THC reached the maximum value, we start fault detection else THC is incremented and rechecked.

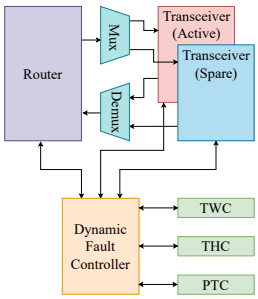


Fig. 2: Spare based Wireless Hub Design

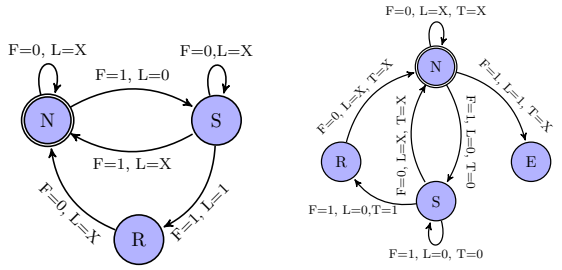
Next, we introduce the method for fault detection. Source hub sends a transceiver fault query (TRFQ) to all the hubs. All hubs send ACK to the source hub after a specified delay. The delay mechanism helps to avoid collision. There may occur two cases. Case 1: If no ACK is received, then two scenarios may occur: (i) Src_Tx faulty or (ii) Src_Rx faulty. If the Src_Tx is faulty, it will not be able to send any query and will not receive ACK. Next, it will not receive any ACK if the Src_Rx is faulty. Case 2: ACK is received from all except one. This reflects that in the destination hub, Dst_Tx or Dst_Rx are faulty. Also, we need to employ a mechanism for the destination hub to understand that it is faulty and recover from the fault. To do so, we use TWC. The TWC will reach its MWT and it will initiate fault detection. It will send a query to every hub. Similarly to the source hub, based on the outcome, the destination hub will deduce that it is faulty.

V. Fault Aware Wireless NoC

This section presents the second contribution of this work. We detail the techniques proposed to mitigate faults at runtime in the transceiver and token controller. A fault aware WH with spare architecture is presented in Fig 2. Each WH has two transceiver section, one active and another spare. A 2:1 mux and demux is used to connect the input and output links of the HC router with the two transceivers. The select line of the mux is controlled by dynamic fault controller (DFC). Also, the individual components are connected with the DFC with controls there functionality.

A. Dynamic Fault Controller

The DFC consists of comparators that compare the value of PTC, THC, and TWC and decide whether a fault has occurred in the system. We use a finite state machine (FSM) for the



(a) Transceiver Fault Recovery Model (b) Token Controller Fault Recovery Model

Fig. 3: Recovery model for different faults in WiNoC

representation of the working of the DFC. We implement separate FSM for transceiver and token controller faults as their fault recovery processes are different. First, we discuss transceiver faults. The FSM for the same is shown in Fig. 3a. In this case, we have 3 states of operation for the WH as follows: (1) Normal (N), (2) Suspend (S), and (3) Recovery (R). We use three variables to decide the changes in the state i.e. fault (F) and local (L). In a fault-free case, the source hub operates in N state (the initial state). However, if it encounters a fault it goes to the S state. If the fault is local (L = 1), it goes to the R state else it remains in the S state. The other hubs which are not participating in the packet communication remain in N state. If the transceiver in the source hub is faulty, the active faulty transceiver is switched off, and the spare one becomes active. It goes back to the N state. Next, we consider fault in the destination hub. The TWC attains its MWT, there exists a fault in the system. Hence, the fault (F) is '1'. The fault status is not known; L = 'X'. The hub goes to S state. It performs the fault detection process and infers that it is faulty. As, F = '1' and L = '1' indicates that the hub goes to R state. After fault recovery, it returns to N state.

Next, we consider the TC faults. We present the recovery model for token controller fault in Fig. 3b. We introduce one new state eliminate (E) and one new variable trigger (T). If the fault is local (L = 1) then the WH is switched off. The TWC in one of other hubs reaches the MWT and initiates the fault detection process. We call this hub as the initiator hub (IH). The IH will know there is fault in the system (F = 1) and move to S state. As soon as the other hubs receives the TCFQ, they halt their TWC counting and move to S state. Depending on the information received, IH will generate a trigger signal (T = 1) and move to R state. Once the fault is recovered, IH sends an update message to all the hubs and moves to N state. All other hubs on receiving the update message update their token ring and move back to N state.

B. Mitigation of transceiver faults

In fault free condition, data communication takes place between the HC router and active transceiver. On occurrence of transceiver fault, the active transceiver is turned off and the spare one takes its place. The spare transceiver is kept switched off during normal operations. Therefore, we use a spare transceiver to handle fault of transmitter or receiver in the active one. Two different issues may occur (1) Packet duplication and (2) Error-free packet transmission.

We discuss the method to address both issues. In the proposed wireless hub, we include two buffers in the transceiver section namely input and output buffer, as presented in Fig. 1b. The input buffer stores the entire packet before it starts packet transmission. The output buffer in the receiver collects the entire packet before it communicates to the wired network. If a fault occurs in the source transceiver, we do not transfer the packet in the buffer of the active transceiver to the spare one. We consider the transmission of the packet failed. During the recovery process, a signal is sent to the source router requesting for retransmission of the packet. At the receiver hub, there may occur two situations. They are as follows (i) there is a part of the packet in the receiver hub, or (ii) no packet in the receiver hub. During recovery, we notify the destination hub about the failure. In the former case, the receiver hub flushes out the packet. The latter does not need such action. This process helps to mitigate the issue of packet duplication and ensure error-free packet transmission.

C. Mitigation of token controller faults

The process to mitigate token controller failure is divided into two parts: (i) token passing and (ii) flow of packets. To resolve the problem regarding flow of packets, we switch off the hub as soon as token controller fault is detected. Packets waiting in the HC router for wireless transmission are redirected towards the destination node using the wired NoC. Also, the system reconfigures itself such that other fault-free hubs pass the token among themselves and operate correctly.

1) Mitigation of token passing problem: When the token controller becomes faulty, the token passing from one hub to another will be disrupted. To solve the above problem, we eject the hub with a faulty token controller out of the ring. If the total number of hubs is N, after ejection we will have N-1 hubs. Again, a Hamiltonian ring is constructed and the token is passed from one hub to another. The hub reconfiguration in case of faults is presented in Fig. 4. We observe that the hub WH_3 is faulty. As WH_3 becomes faulty, it will not be able to release the token. THC will reach maximum value. As soon as THC reaches the threshold, the hub is switched off. All other hubs, that is WH_0, WH_1, and WH_2, after releasing their token started incrementing their TWC. As it is a Hamiltonian ring, the hub WH_0 will reach the MWT first. It will send TCFQ to all the hubs in the WiNoC. WH_1 and WH_2 on receiving the TCQF will halt their TWC and send an ACK to WH_0. However, as WH_3 is switched-off it will not receive the TCFQ and hence will not send an ACK. WH_0 will know the faulty WH. It will generate a self trigger and start the recovery process. Next, the WH_0 sends an update message to all the hubs. This message contains the id of the faulty hub and the source hub. The HAR of the hub containing the address of the faulty hub is replaced by the source hub address. The source hub generates a new token and transfers the token using the newly formed Hamiltonian ring. Next, all the HC routers get updated about the fault information. Depending on

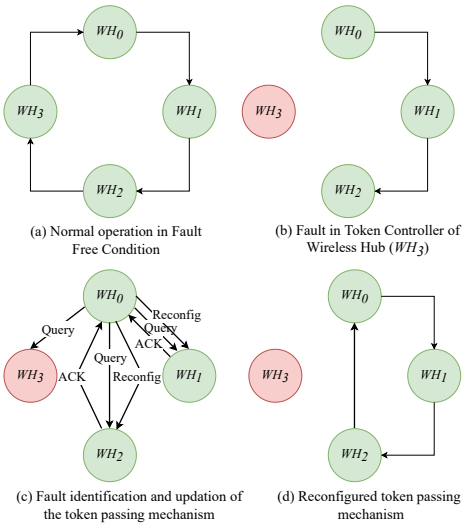


Fig. 4: Mechanism to reconfigure the token passing mechanism in case of fault

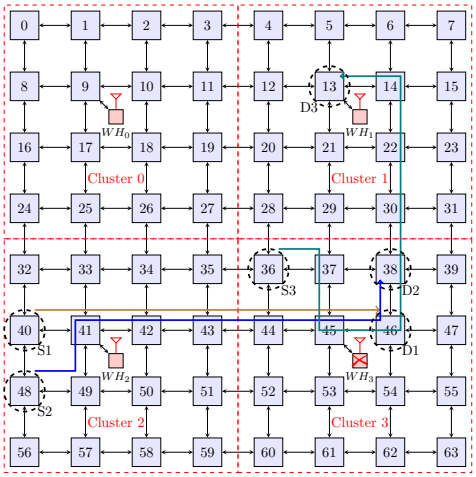


Fig. 5: Example of different fault scenarios

this information, the HC router decides whether the incoming packet should take a wireless route or a wired route to the destination.

2) Detour based Routing: In this scheme, the packets with source destination distance greater than a given threshold are directed towards the HC router. However, if the WH is not available they take the wired route to reach respective destination node. The detour mechanism sometimes uses non-minimal path for packet transmission, which may lead to deadlock. To overcome such situations, we use 2 virtual channels (VC0, and VC1) per physical channel along with turn restriction in our proposed router design. We adopt the following set of rules to avoid deadlock: (i) packets travelling in X direction taking a turn to Y direction remains in the same VC (ii) packets travelling in Y direction taking a turn in X direction should change to a new VC (iii) packets from the lower VC can travel to high VC but packets from higher VC cannot travel to lower VC. In normal fault free condition VC0 is used for packet communication. In case of faults, a detour takes place. We use VC1 for the packets that detour from the HC router (due to fault in WI) to reach the destination.

Let us focus on the example shown in Fig. 5. In this case, we observed that WI_3 is faulty. A packet from source-destination pair S1-D1 communicates using the wireless hubs WH_2 and WH_3. As WH_3 is faulty, the packet gets to know that WH_3 is faulty at the HC router 41 (refer to section V-C1). Therefore, wireless communication is not possible. Thus, it travels through the wired link. For S1-D1, the source and the destination are located on the same row, packets travel along the X-direction without any turn. Therefore no transfer of VC takes place. In the second example of S2-D2, the packet starts from router 48 moves to router 49 along the X-direction. Next, it takes a Y turn and reaches HC router 41. The destination router 37 is in a different row and column. Therefore, the packet has to take an X turn at HC router 41. The packet moves to VC1 following rule 2. Next, the packet takes a Y turn at router 45. Following the above rule, the packet does not shift to a new VC. In example 3, we observe that the packet from S3-D3 takes a non-minimal path to reach the destination. The packet from the source router 36 moves in the X-direction towards the HC router 45. Packets take a Y turn at router 37 and reach the HC router 45. At the HC router, it takes an X turn. Following rule 2, the packet moves to VC1. The packet takes a Y turn at router 46. The packet remains in the same virtual channel VC1 as mentioned in rule 1. Next, the packet takes an X turn at router 6. Here, we follow two rules, that is rules 2 and 3. As the packet is moving through VC1, taking an X turn will not allow it to move back to VC0, due to the fact that VC0 is lower than VC1. Likewise, it moves to VC2. Finally, the packet reaches the digestion router 5. In this work, we consider the wired network to be fault-free. Please note that 'U'-turns are not allowed in the routers, which helps to avoid livelock situations.

VI. Results and Discussion

To analyze the performance of the proposed fault-tolerant wireless NoC, we use the Noxim [18] network-on-chip simulator. Noxim has the mechanism to simulate regular WiNoC architecture. We modified the Noxim simulator to demonstrate the effects of faults and the respective mitigation techniques. In the simulations, we considered two types of faults: (i) transceiver faults, (ii) token controller faults. We injected faults in the WiNoC by disabling the transceiver section or token controller of the randomly selected hub. We used Snipersim 6.1 [19] for full system simulation. We select three SPLASH-2 [20] benchmarks (barnes, fmm, and raytrace) and three PARSEC [21] benchmarks (blackscholes, fluidanimate, and radix), as these represent both communication and computation-intensive workloads. Next, we fed these traces into the Noxim simulator for performance evaluation of WiNoC in terms of network latency and throughput. The Noxim simulator also reports the number of packets transmitted across the WiNoC. We use wormhole routing. In the context of network architecture, the input buffers of the routers are 8-flit long with

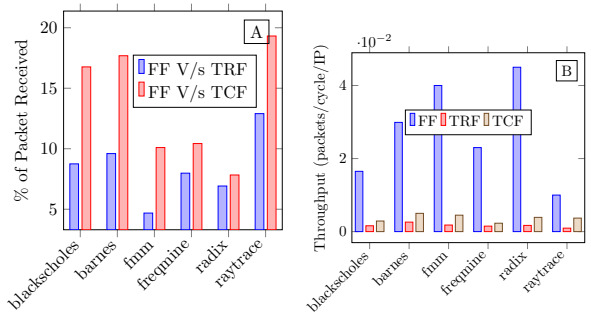


Fig. 6: The effects of TRF and TCF on (A) % of Packet Received and (B) Throughput

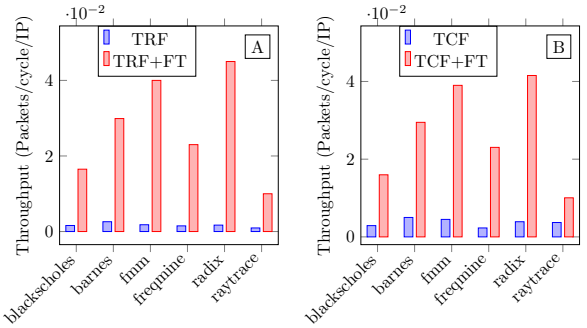


Fig. 7: Throughput using fault-tolerant techniques for (A) TRF and (B) TCF

a buffer depth of 32 bits. Please note that the simulation results given in this paper are conducted on 8 $\times$ 8 with 4 WHs.

A. *Effects of fault on system performance*

In our experiments, we consider failure of one hub in the WiNoC. We implement transceiver fault (TRF) in Noxim simulator by disabling the hub WH_3 in the WiNoC. To implement token controller fault (TCF), we set the maximum token hold time equal to the network simulation time (a worst-case time, which will never be reached) for the hub WH_3. If the transceiver section becomes faulty, a hub will not be able to receive or transmit any packet. Also, the faulty hub will not be able to transfer/receive the token. Therefore, packets awaiting for transmission using the faulty wireless hub will keep waiting and not be served. This will create a congestion in the network. Packets in other hubs will also starve as they will not get the chance for wireless transmission due to the unavailability of the token. This will create back pressure and led to congestion in the entire network. In case of TCF, one of the hub holds the token and all the hubs are kept waiting for the token. This will also lead to back pressure and finally the network will become congested. In Fig. 6(a), we observe a 91.5% and 85.3% drop in received packets for transceiver fault (TRF) and token controller fault (TCF) compared to fault free (FF) operation. This in turn affects the system throughput, where we observe on average 16x and 7x degradation for TRF and TCF compared to FF as shown in Fig. 6(b).

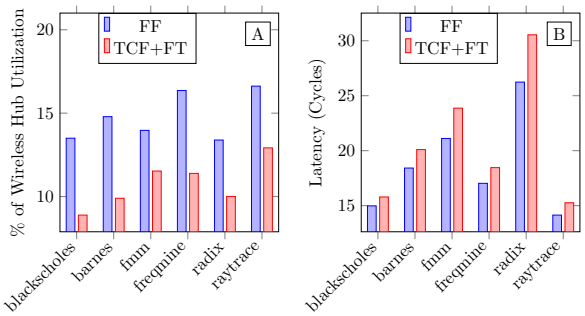


Fig. 8: The effects of fault-tolerant techniques on (A) % of WH utilization and (B) Latency

B. *Evaluation of the proposed approach*

1) Network Throughput: The throughput of the fault-tolerant WiNoC is compared with the WiNoC affected by faults. We see the improvement in Fig. 7. The improvement of throughput for TRF+FT (fault-tolerant) over the faulty case is 16x. The packet loss due to the fault in WH is mitigated by using the spare transceiver. However, in the case of TCF+FT, we observe an average of 6x improvement in throughput compared to TCF as shown in Fig. 7. In the case of the TCF, we eject the faulty hub from the token ring. Therefore, instead of 4 WHs we now have 3 WHs in our WiNoC. Due to this, packets communication to/from the cluster of the faulty hub cannot use the advantage of a single-hop wireless route for long-distance communication.

2) Network Latency: We now focus our attention on Fig 8. We observe that there is a reduction in WH utilization in TCF+FT compared to FF as depicted in Fig 8(a). This is because the hub which encounters token counter faults is switched off. Therefore, those packets which are supposed to communicate through a wireless path are transferred via the wired network resulting in latency degradation. Also, some of the packets have to take a non-minimal route due to the fault (as discussed in section V-C2), which negatively affects the latency. Fig 8(b) shows an average degradation of 10% latency TFC when compared to FF.

3) Area and Power Overhead: In the proposed fault-tolerant WiNoC, every hub contains a spare transceiver section. Thus, we discuss its overhead in terms of power consumption and area. The total power consumption of the transceiver section along with the auxiliary component is 32.3mW [22]. In standby mode, the transceiver section consumes 6.7mW. To reduce this power consumption, we use power gating for the backup transceiver section [15]. In standby mode, its power consumption becomes negligible compared to the active mode. Therefore, the power overhead is minimum. In this work, we use 4 WH for the baseline 8 $\times$ 8 WiNoC. In the case of fault-tolerant WiNoC, we duplicate all components of the transceiver. According to [17], the overhead is less than 6% compared to the regular WiNoC architecture. As the ratio of WH remains low with the NoC size, the fault tolerance provided by the proposed design does not contribute

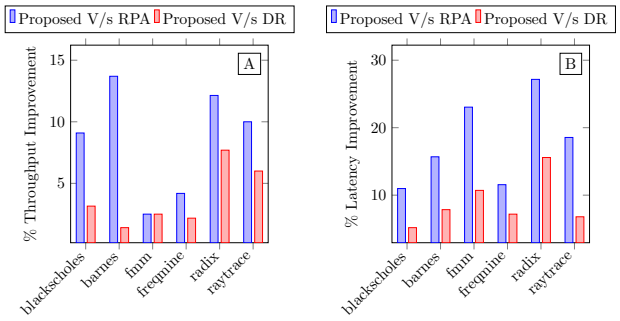


Fig. 9: Comparison with related works in terms of (A) Throughput and (B) Latency

to a large area overhead.

C. Comparison with related works

In previous works [7], [11], each node contains the address of two WI. In case of failure of one WH, packets will move to another WH. Authors in [8] transfer packets to the non-faulty WH in the adjacent cluster. We name all these works as redirect packets to adjacent WH, abbreviated as RPA. The work [12] uses detour routing with a 2-Hop link status checking mechanism to counter wireless link faults. We abbreviate it as DR. All the above works use detour-based routing in case of faults. The detour mechanism will increase the hop count (as the packets will divert from their original shortest path), which will lead to an increase in network latency. To tackle this issue, we use the spare WI in our fault-tolerant WiNoC. Thus, the packets can travel using the original path without detour. We consider single WH failure, i.e., transceiver failure of one WH. We observe that the proposed method shows an average improvement of 17.8% and 8.9% latency improvement compared to RPA and DR. The throughput improvement for the proposed work is (on average) 8.6% and 3.8% compared to RPA and DR. We present the results in Fig 9(a) and 9(b). The improvement against RPA is because the packets from one cluster move to another cluster, thereby increasing the load on the WH belonging to that cluster. Also, these packets communicating to the other cluster may lead to network congestion due to an increase in traffic.

VII. Conclusion

In this paper, a fault-tolerant wireless NoC architecture is proposed, which deals with two types of faults in the wireless hub comprising of the transceiver and token controller failure. We analyze the effects of these faults and propose methods to detect them. To mitigate the former, a spare transceiver is used while the latter is handled using a detour based routing mechanism. We conduct experiments to evaluate the proposed method. We observe that the proposed techniques help to mitigate the faults with minimal degradation in system performance. Also, the proposed work shows improvement in terms of latency and throughput compared with other works in the literature. The future scope of this work is to extend it for larger platform sizes with multi-faults scenarios.

References

[1] C.-H. O. Chen, S. Park *et al.*, "Smart: A single-cycle reconfigurable noc for soc applications," in *2013 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2013, pp. 338–343.

[2] D. Zhu, L. Chen *et al.*, "Application mapping for express channel-based networks-on-chip," in *2014 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2014, pp. 1–6.

[3] A. Karkar, T. Mak *et al.*, "A survey of emerging interconnects for on-chip efficient multicast and broadcast in many-cores," *IEEE Circuits and Systems Magazine*, vol. 16, no. 1, pp. 58–72, 2016.

[4] S. Das, J. R. Doppa *et al.*, "Design-space exploration and optimization of an energy-efficient and reliable 3-d small-world network-on-chip," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 5, pp. 719–732, 2017.

[5] S. Werner, J. Navaridas, and M. Luján, "A survey on optical network-on-chip architectures," *ACM Comput. Surv.*, vol. 50, no. 6, Dec. 2017.

[6] S. Abadal, J. Torrellas *et al.*, "Orthonoc: A broadcast-oriented dual-plane wireless network-on-chip architecture," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 3, pp. 628–641, 2018.

[7] A. Dehghani and K. Jamshidi, "A fault-tolerant hierarchical hybrid mesh-based wireless network-on-chip architecture for multicore platforms," *The Journal of Supercomputing*, vol. 71, no. 8, pp. 3116–3148, Aug 2015.

[8] S. H. Mortazavi, R. Akbar *et al.*, "A fault-tolerant and congestion-aware architecture for wireless networks-on-chip," *Wireless Networks*, vol. 25, no. 6, pp. 3675–3687, Aug 2019.

[9] A. Ganguly, P. Pande *et al.*, "A unified error control coding scheme to enhance the reliability of a hybrid wireless network-on-chip," in *2011 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems*, Oct 2011, pp. 277–285.

[10] A. Vashist, A. Ganguly, and M. Indovina, "Testing winoc-enabled multicore chips with bist for wireless interconnects," in *2018 Twelfth IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*, Oct 2018, pp. 1–8.

[11] A. Dehghani and K. Jamshidi, "A novel approach to optimize fault-tolerant hybrid wireless network-on-chip architectures," *J. Emerg. Technol. Comput. Syst.*, vol. 12, no. 4, pp. 45:1–45:37, Mar. 2016.

[12] Y. Ouyang, Q. Wang *et al.*, "A novel low-latency regional fault-aware fault-tolerant routing algorithm for wireless noc," *IEEE Access*, vol. 8, pp. 22 650–22 663, 2020.

[13] A. Ganguly, P. Wettin *et al.*, "Complex network inspired fault-tolerant noc architectures with wireless links," in *Proceedings of the Fifth ACM/IEEE International Symposium*, May 2011, pp. 169–176.

[14] C. Wang, W.-H. Hu, and N. Bagherzadeh, "A wireless network-on-chip design for multicore platforms," in *2011 19th International Euromicro Conference on Parallel, Distributed and Network-Based Processing*, 2011, pp. 409–416.

[15] H. K. Mondal, R. C. Cataldo *et al.*, "Broadcast- and power-aware wireless noc for barrier synchronization in parallel computing," in *2018 31st IEEE Int. System-on-Chip Conference (SOCC)*, Sep. 2018, pp. 1–6.

[16] B. A. Floyd, C.-M. Hung, and K. K. O, "Intra-chip wireless interconnect for clock distribution implemented with integrated antennas, receivers, and transmitters," *IEEE Journal of Solid-State Circuits*, vol. 37, no. 5, pp. 543–552, May 2002.

[17] S. Deb, K. Chang *et al.*, "Design of an energy-efficient cmos-compatible noc architecture with millimeter-wave wireless interconnects," *IEEE Transactions on Computers*, vol. 62, no. 12, pp. 2382–2396, Dec 2013.

[18] V. Catania, A. Mineo *et al.*, "Improving energy efficiency in wireless network-on-chip architectures," *J. Emerg. Technol. Comput. Syst.*, vol. 14, no. 1, pp. 9:1–9:24, Nov. 2017.

[19] T. E. Carlson, W. Heirman *et al.*, "An evaluation of high-level mechanistic core models," *ACM Transactions on Architecture and Code Optimization (TACO)*, 2014.

[20] S. C. Woo, M. Ohara *et al.*, "The splash-2 programs: Characterization and methodological considerations," *SIGARCH Comput. Archit. News*, vol. 23, no. 2, pp. 24–36, May 1995.

[21] C. Bienia and K. Li, "Parsec 2.0: A new benchmark suite for chip-multiprocessors," in *Proceedings of the 5th Annual Workshop on Modeling, Benchmarking and Simulation*, June 2009.

[22] X. Yu, J. Baylon *et al.*, "Architecture and design of multichannel millimeter-wave wireless noc," *IEEE Design Test*, vol. 31, no. 6, pp. 19–28, Dec 2014.