2023 3rd International Conference on Smart Generation Computing, Communication and Networking (SMART GENCON)
Karnataka, India. Dec 29-31, 2023

2023 3rd International Conference on Smart Generation Computing, Communication and Networking (SMART GENCON) | 979-8-3503-1912-5/23/$31.00 ©2023 IEEE | DOI: 10.1109/SMARTGENCON60755.2023.10442582

Adaptive Time-Triggered Network-on-Chip Architecture: Enhancing Safety

Rakotojaona Nambinina, Veit Wiese, Pascal Muoka, Daniel Onwuchekwa, Roman Obermaisser
Chair for Embedded Systems
University of Siegen
Siegen, Germany
andrianoelisoa.rakotojaona@uni-siegen.de

***Abstract*—Real-time computing systems are designed to meet strict timing constraints and respond to events or inputs within specific deadlines. These systems are commonly used in safety-critical applications such as spacecraft, medical devices, industrial control, and automotive systems. Engineers rely on various scheduling techniques to ensure that timing constraints are met. One such technique is static allocation in time-triggered systems. This technique offers valuable advantages in terms of system safety and dependability. It enables efficient resource usage in Network-on-Chip (NoC) architectures by minimizing message congestion and contention. However, its static nature can present limitations in terms of fault tolerance. This paper focuses on developing and evaluating fault tolerance techniques tailored for NoC-based multi-core architectures to enhance safety. It introduces adaptation techniques that tolerate permanent faults within the time-triggered NoC architecture. Additionally, the paper outlines the incorporation of seamless redundancy mechanisms at the network interface (NI) level, specifically designed to tolerate transient faults that may occur within NoC components, such as routers and links. These mechanisms play a crucial role in safeguarding critical data, further enhancing the reliability and safety of real-time systems in safety-critical applications.**

***Index Terms*—Network-on-Chip, Time-triggered, Adaptation, Seamless Redundancy, Multi-core Architecture.**

I. Introduction

In 1965, Moore projected that the transistors within an integrated circuit would double roughly every 18 months. Remarkably, the semiconductor industry has managed to keep up with this prediction, integrating up to billions of transistors on a single chip. This has made it possible to integrate multiple cores on a single chip. However, as the number of cores integrated into System-on-Chips (SoCs) increases, using a shared bus for communication between cores in SoCs becomes inefficient. Communication between cores is essential to ensure high performance and meet real-time requirements. The limitations of shared bus architectures, such as limited bandwidth, scalability challenges, lack of flexibility, non-deterministic communication, and vulnerability to single points of failure, have led to the emergence of NoCs as a new communication paradigm [1]. NoCs provide a packet-switched communication network with multiple paths for various communication flows, thereby addressing the inefficiencies of shared bus architectures. In safety-critical applications, like automotive and avionics systems, deterministic communication is crucial to ensure predictable and timely transmission and reception of messages. This predictability is vital for meeting real-time requirements and maintaining timing guarantees, which is essential in real-time applications. Time-triggered communication plays a critical role in fulfilling these real-time demands. However, using static allocation in time-triggered communication presents challenges when dealing with dynamic workloads and evolving system requirements, necessitating a more flexible approach. Additionally, it poses difficulties in achieving fault tolerance, which is crucial for ensuring safety in safety-critical systems. To overcome these challenges, we introduce the Adaptive Time-Triggered Network-on-Chip (ATTNoC) architecture designed for multi-core systems to enhance the safety of NoC-based multi-core architecture. This architecture extends the functionality of an event-triggered-based NoC called LISNoC [2] to support time-triggered communication. Additionally, the architecture incorporates fault tolerance mechanisms, such as adaptation and redundancy, to enhance the system's safety in the presence of permanent and transient faults. The ATTNoC supports multiple schedules, allowing the NoC to switch between schedules in response to context events, such as permanent faults in routers, links, NIs, or cores. This enhances safety by isolating faulty sub-components and redistributing tasks or messages to other available resources. This ensures the system's communication is not interrupted when a permanent fault occurs in an NoC [3]. Moreover, seamless redundancy is employed to enhance further the ATTNoC's safety, where critical messages can be transmitted via dual channels to tolerate permanent or transient faults in the routers or links. This approach eliminates the need for fully duplicating NoC resources, reducing overhead.

The remainder of this work is structured as follows: Section II discusses the related works. Section III gives an overview of the Adaptive Time-Triggered Network-on-Chip Architecture. An experimental setup and results evaluation are presented in section IV. Finally, the conclusion is discussed in Section V.

II. Related works for Fault Tolerance Techniques for NoC Architecture

Fault tolerance in NoC architectures has made significant progress in detecting, correcting, and masking faults. One commonly used approach is spatial redundancy, where essential NoC components are replicated to achieve adequate fault tolerance. Spatial redundancy can be classified into three

categories: static, dynamic, and hybrid redundancy [4]. Static redundancy passively masks faults, while dynamic redundancy detects faults and takes corrective action. Hybrid redundancy combines both static and dynamic methods [5]. For example, triple modular redundancy (TMR) is a well-known static redundancy technique that can mask a single fault but requires significant additional hardware [4]. Redundancy can be applied at various layers of an NoC, including links, routers, NIs, or cores. At the data link layer, techniques like TMR and spare wires are widely employed to ensure the proper functionality of link lines and control signals within the NoC. TMR protects critical control signals, such as the NACK signal in hop-to-hop communication [6], [7], and logic elements like the crossbar multiplexer, ensuring correct switching even in the presence of faults [8]. To enhance NoC connectivity and ensure uninterrupted communication even in the presence of faulty links, Kakoee et al. [9] proposed a method that duplicates all physical links between routers and incorporates dynamic redundancy. This approach involves replacing detected faulty links with spare links. The number of spare links used determines the fault tolerance capability of the system. However, reliable fault detection methods are crucial for effective dynamic fault tolerance to prevent undetected faults from compromising system safety. Techniques such as Hamming code, configuration vectors with tristate gates, or Built-In Self Test (BIST) can be employed for fault detection. While spatial redundancy provides significant fault tolerance capacity, it often comes with considerable overhead. To reduce this overhead, two trade-offs need to be made. First, the granularity of redundancy should be carefully determined to optimize costs. For example, in the case of TMR, the cost heavily depends on the partition granularity [10]. The second trade-off is combining spatial redundancy with other techniques to reduce overall costs. The system can enhance fault tolerance by integrating spatial redundancy with complementary fault tolerance techniques, taking into account the associated overhead. This approach enables efficient resource allocation and balances the trade-off between fault tolerance and cost efficiency.

In addition to spatial redundancy, it is essential to consider additional techniques to ensure the reliability of the entire NoC platform. NoCs are susceptible to various faults, including transient and intermittent faults [11]. Fault recovery is the ultimate goal of fault-tolerant systems. Error detection codes (EDC), error correction codes (ECC), and advanced coding schemes such as Reed-Solomon codes can be used to detect and correct a transient fault, such as bit flips, which occur due to low noise margins, electromagnetic coupling effects, or crosstalk [12], [13], [14], [15]. Temporal redundancy, such as retransmission at the link-to-link or end-to-end level at different time intervals, is one approach for enhancing fault tolerance by retransmitting data during a transient fault [14]. This method can improve fault tolerance by introducing redundancy in data transmission. On the other hand, traditional end-to-end retransmission introduces significant packet latency and requires additional infrastructure support, such as an ack/nack protocol. To mitigate these drawbacks, a hop-by-hop retransmission scheme is often used. This scheme can reduce overall end-to-end latency by handling retransmission locally at each hop rather than relying on acknowledgments from the destination, as in the case of traditional end-to-end retransmission. Additionally, detecting and tolerating faults in the control path of routers pose significant challenges. Such faults can lead to errors in the routing algorithm and result in incorrect establishment of connections between input and output ports. Therefore, it is crucial to develop and implement effective strategies that ensure the reliable functioning of the network and mitigate potential disruptions. One standard solution involves disabling faulty components and using an appropriate routing algorithm to bypass disabled links, routers, or both. However, many of these methods require halting network operations and resetting the system. The new topology is discovered during the setup phase, and the network can resume functioning under the new configuration. Some proposals aim to avoid the costly system reset [15], [16], but they may result in packet loss during the transition phase. Nevertheless, these proposals protect the network from entering deadlock situations without needing a reset. Several approaches, including architectural solutions and retransmission techniques, have been proposed and combined in architecture to increase network reliability against transient faults [17]. One such proposal, Bulletproof [18], integrates techniques like triple modular redundancy, end-to-end fault detection, and resource sparing at different levels (system, component, and gate levels). It explores the trade-offs between area, power, latency, and reliability, but a comprehensive approach that balances various trade-offs is required.

The fault tolerance techniques presented in this work contribute to the field of fault tolerance in NoCs by offering several key advantages over existing state-of-the-art approaches. The contributions of this work can be summarized as follows: Dynamic schedule reconfiguration: Instead of relying on a fixed schedule commonly used in NoC architectures, this work introduces dynamic reconfiguration of the NoC schedule based on context events. This approach allows the system to use multiple schedules and adaptively reconfigure the schedule based on context events, making the system more flexible and adaptable. By responding to changing conditions, the system enhances its fault tolerance capabilities. Seamless redundancy integration at the NI level: This mechanism enables the NI to duplicate critical data and transmit it over different paths, allowing the system to tolerate transient and permanent faults in the routers and links during message exchanges. Since only the critical data is duplicated within NIs, there is no need to duplicate all messages within the NoC, resulting in lower overhead. The time-triggered communication mechanism in the NoC reduces the risk of message collisions and delays, ensuring that messages are transmitted within a specific time. This mechanism is crucial in real-time applications where timely message delivery is paramount. By enhancing the reliability of message delivery, the system optimizes performance, leading to better overall system efficiency. By incorporating seamless redundancy mechanisms, time-triggered communication, and adaptation techniques, the presented approaches contribute to

fault tolerance in the NoC domain. They provide valuable insights into improving network reliability, adaptability, and communication efficiency, furthering the understanding and implementation of fault tolerance in NoC architectures.

III. Adaptive Time-Triggered Network-on-Chip Architecture

The Adaptive Time-Triggered Network-on-Chip (ATTNoC) is an NoC-based multi-core architecture designed with fault tolerance techniques to enhance the safety of such multi-core systems. The ATTNoC integrates several services, including time-triggered communication, adaptability, and redundancy mechanisms. These services are described in the following items.

- Time-triggered communication: This service facilitates the exchange of messages within the NoC based on a pre-defined schedule, ensuring consistent and predictable communication. By adhering to the NoC schedule, each message and task can access resources at predetermined times, preventing message collision and reducing the risk of missed deadlines that can negatively impact real-time applications.
- Fault tolerance through adaptation: The NI in ATTNoC architectures supports multiple schedules, allowing it to switch between schedules in response to context events like permanent faults in NoC resources such as NIs, routers, cores, and links. This ensures that the systems continue to operate despite faults in the NoC resources. This approach aims to preserve the crucial time-triggered system properties, including implicit synchronization and avoidance of resource contention [3].
- Seamless redundancy mechanisms: The ATTNoC architecture introduces two time-triggered NI to support mixed-criticality: Safety Critical NI (SCNI) and Non-Safety Critical NI (NSCNI). SCNI has two modes of operation: redundant and non-redundant. In redundant mode, SCNI transmits critical messages over a dual channel, while non-critical messages use a single channel to optimize resource usage. NSCNI only supports non-redundant mode.

The ATTNoC architecture builds upon the open-source event-triggered LISNoC presented in Section III-A. It supports topologies like meshes and connects resources like hard and soft processors, memory subsystems, etc. Figure 1 gives an overview of the ATTNoC architecture. It consists of tiles interconnected through an NoC. The NoC comprises routers connected to other routers and tiles via communication links. Each tile consists of three parts: cores for application services, an adaptation unit (indicated by the blue area in Figure 1) composed of four elements: context monitor, context agreement, schedule memory, and time-triggered dispatcher to switch between schedules during context events, and SCNI and NSCNI to access the NoC. Furthermore, the platform services employed in the ATTNoC are illustrated on the right side of Figure 1. These services encompass global time-based functionality, serving as a time reference within the ATTNoC. Adaptation techniques enable the NoC to reconfigure its schedule. Seamless redundancy allows the NI to duplicate and transfer critical data through a dual channel. Lastly, time-triggered communication enables the NoC to inject messages into the NoC based on a predefined schedule.

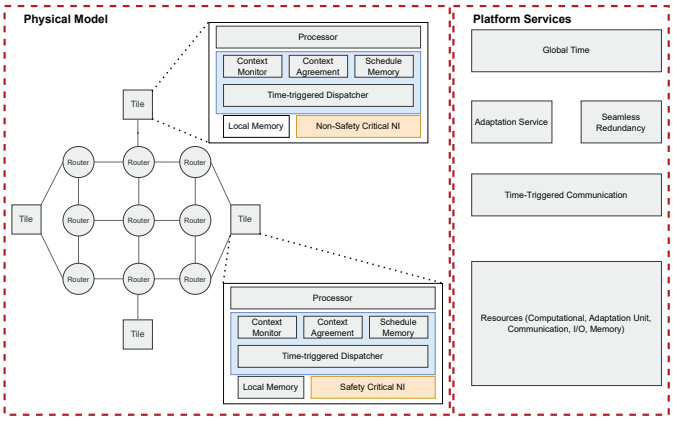


Fig. 1. System Model of an Adaptive Time-Triggered Network-on-Chip

A. LISNoC

LISNoC is an event-triggered-based NoC. It is an open-source NoC implemented in Verilog, primarily used for academic purposes [2]. It serves as the foundation for implementing ATTNoC, which incorporates fault tolerance techniques. Some of the main features of LISNoC include virtual channel support, flexible configuration, wormhole routing, and round-robin arbitration [2]. The LISNoC uses a packet format for data transmission, dividing each packet into flits. The header of each packet contains all the necessary information about the destination processing element. Moreover, the LISNoC has expanded to include source-based routing [19], time-triggered communication, adaptability, and redundancy mechanisms to support predictable communication and real-time applications [20].

B. Adaptation Unit

The adaptation unit in ATTNoC serves as a critical component designed to facilitate schedule switching and manage context events within the ATTNoC. These context events may include permanent faults that have occurred in the NoC resources, such as routers, links, and cores. This unit comprises four key elements: Context monitor: This component is responsible for gathering and reporting adaptation-relevant data to the context agreement. It accomplishes this by continuously monitoring the current context values from adjacent cores. Context agreement: Its role is to exchange the context information provided by the context monitor among all distributed adaptation units. It does so through the usage of the triple ring structure. This ensures that all adaptation units are well-informed about local context events happening in other context agreements, and can tolerate faults, such as message

corruption, during the exchange of context information. Schedule memory: This is a storage unit that retains precomputed schedules. Time-triggered dispatcher: It is responsible for initiating the operation of the context monitor, context agreement, and schedule switching within the ATTNoC. Its purpose is to enable synchronized schedule switching across the distributed NI in ATTNoC [3], [21].

C. Fault Model for ATTNoC Architecture

A fault model is a method used to identify potential faults in a system. Faults not considered in the model are guaranteed not to be covered by the fault tolerance techniques used in the ATTNoC.

1) Failure Mode: We simulate permanent and transient faults in the ATTNoC components, such as routers, links, NIs, and cores. Therefore, the fault tolerance techniques employed in the ATTNoC must be resilient against failures of cores, links, routers, and NIs.

2) Fault Containment Region: The fault containment regions (FCRs) are where faults may occur at run-time and are tolerated. It is the delimiter of the immediate impact of a fault and is defined to cover the cores, links, and NIs, as shown in Figure 2. No faults that originate within an FCR should negatively impact another FCR. This means that in case of permanent faults, the core should still be able to compute tasks and exchange messages to maintain functionality. Similarly, in the event of transient faults in routers or links during the exchange of critical messages, the NI receiver should still receive the messages. Table I below shows how each

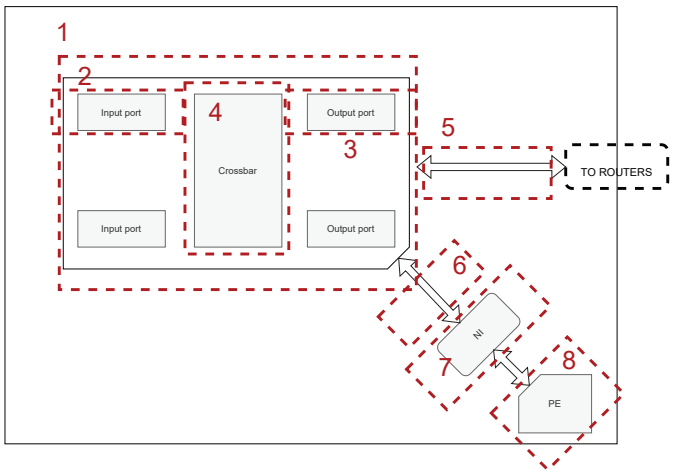


Fig. 2. FCR in ATTNoC Architecture

permanent fault corresponds to a fault type in high-level abstraction.

3) Fault Assumptions: We defined our FCR to cover NIs, routers, links, and cores. This means we accept that a fault can happen with this FCR but may not immediately impact the other FCRs. Here, we consider three common faults that can occur during run-time in the NoC and cause the system to enter a faulty state. The first fault is a delay, which can be critical

TABLE I
PERMANENT FAULT IN ATTNOC

Component ID	Location
1	Router fault
2	Input router fault (FIFO)
3	Output router fault (FIFO)
4	Crossbar fault
5	Router-Router link fault
6	Router-NI link fault
7	NI fault
8	Processing element fault

in real-time applications where message delays can cause the system to miss its deadline. The faulty component responsible for this delay slows down any message that passes through the faulty component, often in the routers, NIs, links, or cores. The second fault we consider is message corruption. In this case, the faulty component corrupts any message that passes through the faulty component. The third fault we consider is an open circuit in the ATTNoC, where the data passing the faulty component is always lost. The adaptation features implemented in the ATTNoC are designed to tolerate and accommodate such delays, message corruption, and dropped messages that persist throughout the system's lifetime, eventually leading to system failure. However, it is essential to note that the adaptation features within the ATTNoC are not intended to tolerate transient faults, as these faults can be rapidly recovered within a few clock cycles. Therefore, there is no need for the systems to reconfigure their schedules for transient faults. Nevertheless, the redundancy mechanisms implemented in the SCNI are specifically designed to tolerate transient faults in the form of data corruption that may occur in the routers and links of the NoC during the exchange of critical messages between SCNIs. This is achieved through redundant transmissions by duplicating the critical messages and transmitting them through dual channels in the NoC.

D. Adaptation in ATTNoC Architecture

The adaptation techniques employed in the ATTNoC allow for tolerating permanent faults that may occur within the NoC during run-time. By reconfiguring the ATTNoC with a new schedule and isolating the faulty component, these techniques enable the system to continue functioning effectively. An example scenario in Figure 3 illustrates the need for schedule switching due to a permanent fault. In this scenario, router R4 is assumed to be faulty, causing any messages passing through it to fail to reach their intended destinations. To overcome this fault and ensure proper communication, the ATTNoC must undergo a schedule change. The failed router R4 is removed in the new schedule, and all messages that pass through R4 are rerouted to alternative paths, as demonstrated in Figure 3. Through this adaptation process, the ATTNoC effectively addresses the permanent fault by adjusting its schedule and rerouting the messages to isolate the faulty component. By isolating the faulty component and reconfiguring the system

accordingly, the ATTNoC maintains its functionality and ensures uninterrupted communication within the network.

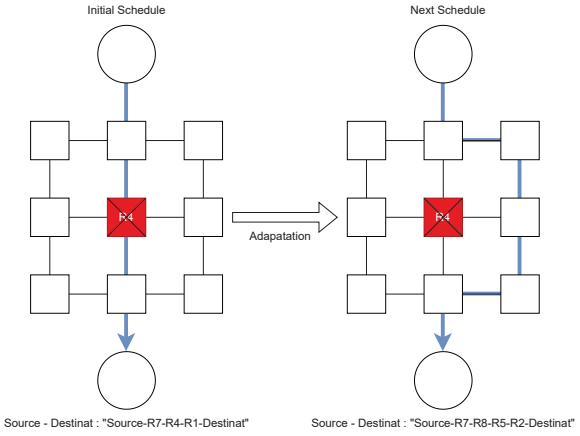


Fig. 3. Adaptation by Changing Schedule and Isolating a Faulty Router in ATTNoC

E. Seamless Redundancy Mechanisms in ATTNoC Architecture

The SCNI is an interface that connects the core with the routers and incorporates a seamless redundancy mechanism, which allows the NI to transfer data through a dual channel, depending on the criticality of messages. The architecture of SCNI is depicted in Figure 4. The SCNI uses dual packetization and depacketization modules to support redundant message transmission. These modules are connected to separate routers that allow the SCNI to transmit duplicate messages over two paths within the NoC. The redundancy controller manages the redundancy mode operation. The redundancy controller duplicates messages from the core interface when the redundant mode is required. Otherwise, it activates one of the packetization blocks for normal transmission. On the receiver side, the redundancy controller selects messages from both depacketization modules based on the order in which they were received and their integrity, as determined by a CRC (Cyclic Redundancy Check). Following is the description of each block that comprises the SCNI.

- The core interface facilitates access to the NI and, consequently, the NoC through ports and registers. Ports establish a memory-mapped interface between the core and the NoC and temporarily store messages until they are either transmitted to the NoC or retrieved by the core.
- Adaptive time-triggered dispatcher is the only NI element that interprets the schedule and triggers the periodic transmission of the time-triggered messages. It also ensures the NI's adaptation by reconfiguring the schedule with a new schedule when a context event occurs.
- Packetization and depacketization are crucial processes in the communication flow between the core interface and the router within the NI. Packetization involves encoding messages from the core interface before transmitting them to the router. This encoding step prepares the messages to be efficiently transmitted over the NoC. In addition, it involves organizing the messages into packets, smaller data units that can be easily transmitted through the network. On the other hand, depacketization is receiving messages from the NoC router and decoding them before writing them back to the core interface. This step reverses the packetization process and prepares the messages for further processing by the connected processing element.
- The router interface is an interface that connects both the router and NI. The router and NI use a handshaking protocol to transfer data from the router to NI or vice versa.
- Memory store path: Since source-based routing is used in the ATTNoC, the route for each message is predetermined and stored in a memory called the memory store path within the NI. When packetization forms a packet, the message's path is retrieved from the memory and added to the head flit of each packet.
- The redundancy sender controller plays a crucial role in determining the redundancy mode operation of the SCNI. It decides whether the NI should transmit data redundantly or non-redundantly. During the design phase, each output port of the core interface is pre-configured to indicate its intended transmission mode. The triggering of output ports for message injection in the NoC follows a predefined schedule set by the time-triggered dispatcher. When the output port of the core interface is triggered to inject messages, the redundancy sender controller receives data from the core interface along with a three-bit controller signal. This signal helps determine the redundancy mode operation of the SCNI. The behavior of the redundancy sender controller is summarized in Table II. For instance, when the control bits are "100," it indicates that messages from the output port of the core interface originate from a redundant port. In this case, the redundancy sender controller duplicates the messages and transmits them through both packetization blocks using a dual channel. If the control bits are "010," only the first

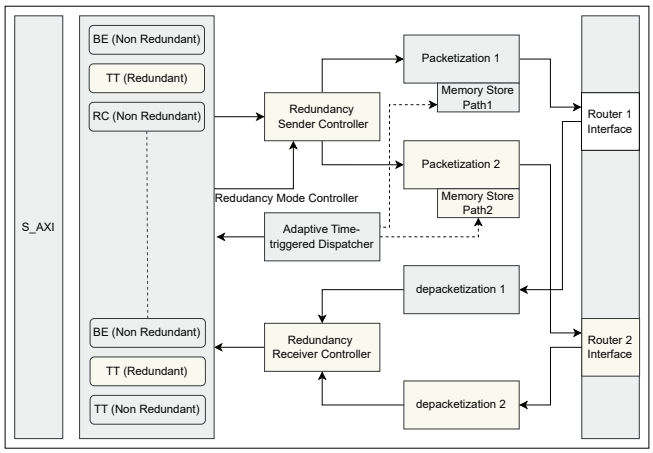


Fig. 4. Safety Critical NI Architecture

packetization block receives data, and the second packetization block remains inactive. Consequently, transmission is exclusively handled by the first packetization block. Similarly, when the control bits are set to '001,' the second packetization block receives data while the first packetization block remains inactive. In this scenario, the second packetization block manages the transfer of flits (flow control units).

TABLE II
REDUNDANCY SENDER CONTROLLER OPERATION

Control bit of redundancy controller	Seamless redundancy operation	Activated packetization
"100"	Seamless redundancy mode	Both packetization are activated
"010"	Non-redundant	Packetization 1 is only activated
"001"	Non-redundant	Packetization 2 is only activated

- The redundancy receiver controller in the SCNI is responsible for selecting the correct data from two depacketization blocks based on data correctness and the sequence number of messages. The redundancy receiver controller performs several checks upon receiving data from the depacketization blocks. First, it uses the sequence number of the packet's head flits to identify the corresponding data from the dual channels. Next, it performs a CRC (Cyclic Redundancy Check) on the received data to determine its correctness. The redundancy receiver controller maintains a status register that stores the correctness status of the received data. A value of "10" indicates that the data is not corrupted, while a value of "01" signifies data corruption. Based on the status of the data, the redundancy receiver controller decides which data to accept. The decision process is described in Table III. If both data from the depacketization processes are not corrupted, the redundancy receiver controller accepts the data from the first depacketization process and discards the data from the second process. When one of the data is corrupted, the redundancy receiver controller accepts the non-corrupted data and discards the corrupted data from one of the depacketization blocks.

TABLE III
REDUNDANCY RECEIVER DATA SELECTION

Status Depacketization 1	Status Depacketization 2	Selected data from
"01"	"01"	Dropped both data
"01"	"10"	Depacketization 2
"10"	"01"	Depacketization 1
"10"	"10"	Depacketization 1
"10"	"x"	Depacketization 1
"x"	"10"	Depacketization 2

IV. EXPERIMENT SETUP AND RESULTS EVALUATION

This experiment aimed to comprehensively assess the effectiveness of the fault tolerance techniques used in the ATTNoC, focusing on adaptation and seamless redundancy in tolerating both permanent and transient faults that may occur in the NoC resources, such as routers, NIs, and links. A test case approach was employed to achieve this assessment, intentionally introducing various types of faults, including permanent and non-permanent ones, into the NoC resources. These faults were designed to create scenarios such as delays, lost messages, and corrupted messages within the faulty NoC resources. By adopting this approach, the experiment established a controlled environment to evaluate the system's resilience and effectiveness in mitigating the impact of the introduced faults. Following the predefined test case approach, a subsequent testing phase took a different approach. Randomized faults in the form of transient faults through message corruption were introduced in the NoC to evaluate the effectiveness of seamless redundancy in the ATTNoC when exchanging critical messages.

A. Experimental Setup based on Permanent and Transient Faults

In this experimental setup, a 3x3 mesh ATTNoC architecture was designed using Vivado Xilinx and subjected to various fault conditions, including delay, message corruption, and open link, as depicted in Figure 6. The study used a fault model with four parameters: F = {StartTime, Duration, Location, Fault-Type}, where the StartTime indicates when the fault occurs, duration represents how long the fault persists, location indicates where the fault occurs, and Fault-type represents the type of fault. These parameters were varied to evaluate the system's reliability under different fault conditions. In addition, the total number of uncorrupted received packets was used to assess the system's ability to tolerate a fault when fault tolerance techniques were used. The experimental setup allowed for assessing the effectiveness of the adaptation techniques and redundancy built into the ATTNoC architecture in tolerating faults, providing valuable insights into its reliability under different conditions. The fault scenario presented in Figures 5 and 6 served as the basis for evaluating the functional behavior of the ATTNoC system under diverse fault conditions encompassing three distinct fault types, namely F1, F2, and F3. It's important to note that each fault scenario can encompass multiple individual faults, represented as S = {F1,..., Fn}. The following points provide a detailed description of the fault scenarios employed in our experiments.

- The fault denoted as F1 represents a permanent fault at the network link connecting R6 to NI2. This fault leads to message corruption. The fault is activated when the time reaches 600 times the period, equal to 600 * T with T = 488.28 μs. The period here refers to the period used to establish time-triggered communication. This means eight packets are exchanged between the NoC every period, as depicted in Figure 6. The term "permanent" signifies that

the fault is persistent and does not resolve independently, requiring maintenance or intervention to rectify it.

- The fault denoted as F2 is a non-permanent fault at router R7 in the network. This fault introduces a delay of 10 clock cycles for each message passing through the affected router. The fault is activated when the time reaches 250 times the period. It remains active for a single period, after which the faulty link is automatically recovered.
- The fault known as F3 is a non-permanent fault that occurs at the link connecting routers R1 and R2 in the NoC. This fault results in the dropping of flits transmitted through this link. The fault is activated when the time reaches 300 times the period. It remains active for a single period, after which the faulty link is automatically recovered.

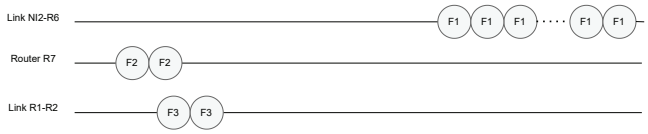
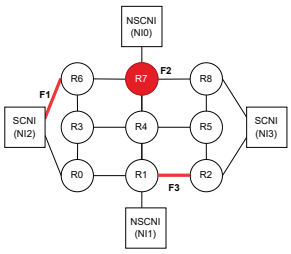


Fig. 5. Fault Scenario S= {F1,F2,F3} and Communication Schedule in the ATTNoC.

Figure 6 illustrates the fault scenario (S = {F1, F2, F3}) in the ATTNoC architecture, where the routers and links shown in red indicate faulty NoC components. The table on the right side of the figure presents the communication schedules used in the experiment. Schedule 0 represents the initial schedule, while schedule 1 is the precomputed schedule to which the system can switch when a permanent fault occurs in the link connecting NI2 and router R6. This configuration was designed to simulate a scenario involving faulty communication components within the network.



NI		Schedule 0	Schedule 1 (Permanent fault on link NI2-R6)
NI (Source)	NI (Sink)	Path	Path
0	1	R7-R4-R1	R7-R4-R1
1	0	R1-R4-R7	R1-R4-R7
2	3	R6-R7-R8/ Redundant (R0-R1-R2))	R6-R7-R8/ Redundant (R0-R1-R2))
3	2	R8-R7-R6/ Redundant (R2-R1-R0)	R8-R7-R6/ Redundant (R2-R1-R0)
2	1	R0-R1	R0-R1
2	0	R6-R7	R0-R1-R4-R7
3	1	R2-R1	R2-R1
0	2	R7-R6	R7-R4-R3-R0

Fig. 6. Fault Scenario S= {F1,F2,F3}. Red in the Architecture Highlights an Error.

To evaluate the effectiveness of fault tolerance techniques used in the ATTNoC, a comparative analysis was conducted between different configurations, some with fault tolerance techniques such as redundancy and adaptation mechanisms and others without. The ATTNoC configuration used for this experiment includes four NIs that facilitate the exchange of eight packets in a period equal to 488.28 μs. The experimental setup involved exchanging 8,000 packets for evaluation, focusing on the fault scenario denoted as S={F1, F2, F3}.

B. Results and Discussion

Figure 7 presents the total number of packets received between the cores for four cases. The x-axis represents the four cases, while the y-axis displays the total number of packets received for each case. Case 1 represented the ATTNoC operating without any faults and received the highest number of packets at 8000, serving as a reference for the system's optimal performance under normal conditions. Cases 2-4 depict the ATTNoC with faults, resulting in fewer packets received. Specifically, Case 2 shows the ATTNoC without fault tolerance techniques, where faults were introduced, leading to the lowest number of packets received at 6399. This indicates that faults can significantly impact the system's reliability, resulting in a considerable loss of packets. In Case 3, the ATTNoC has redundancy features but no adaptation features. It received 7203 packets, higher than the ATTNoC without redundancy and adaptation features. This indicates that redundancy mechanisms can be useful in reducing the impact of faults. Case 4 presents the ATTNoC with adaptation features and redundancy mechanisms, receiving the highest number of packets at 7994, particularly when Fault S = {F1, F2, F3} was injected. This result highlights the effectiveness of incorporating redundancy and adaptation mechanisms in ATTNoC design to enhance its reliability in the event of transient and permanent faults. The comparison results confirm the significance of redundancy and adaptation features when designing an ATTNoC architecture. These features enhance system reliability in the presence of faults, underlining their significance in ATTNoC design, particularly for safety-critical systems. However, incorporating fault tolerance techniques such as redundancy and adaptation in the ATTNoC can increase resource usage in the NoC, resulting in high overheads as it requires extra resources on the hardware.

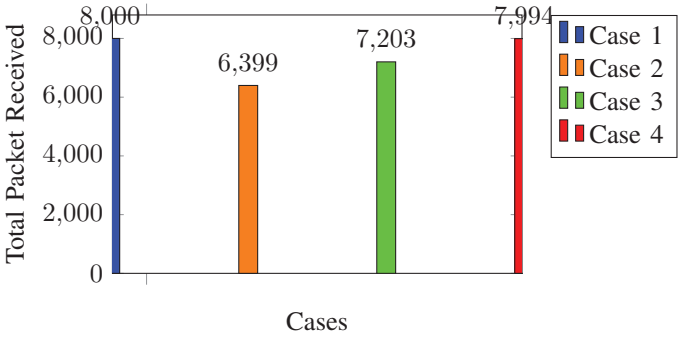


Fig. 7. Comparison of the Total Packets Received in four Cases of ATTNoC Communication.

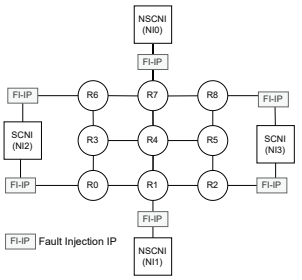
C. Experiment Setup Based on Randomized Test Case

An intellectual property (IP) for fault injection, as proposed by [22], was incorporated into the ATTNoC experimental setup. It aimed to introduce transient faults into the NoC resources, as Figure 8 depicts. The goal was to assess the

effectiveness of redundancy mechanisms employed in the ATTNoC in tolerating such faults. These transient faults occurred sporadically and recovered within a few clock cycles. The fault injection IP was positioned between the router and the NI, allowing it to corrupt messages exchanged within the ATTNoC, potentially leading to message corruption. To achieve its purpose, the fault injection IP used two Linear Feedback Shift Registers (LFSRs) and a third LFSR, as shown in Figure 9. The first two LFSRs were used as pseudo-random number generators. The specified number of bits from both LFSRs were compared. When these specific bits were equal, the fault injection IP could corrupt the flits passing through the links, potentially resulting in corrupted messages, misrouted data, and congestion within the NoC. The third LFSR, which was 6 bits in size, determined the specific bit of the flits where the corruption occurred. Since the data flit was only 34 bits, values higher than 34 were considered invalid, and no corruption was injected. Thus, values ranging from 0 to 33 were effectively used to determine the position of the corruption within the 34-bit flit. Two of these bits indicated the type of flits (head, body, or last), while the remaining bits stored the actual data or opcode. The fault injection rate could be adjusted by modifying the number of matching bits to compare the two output data from the two LFSRs. More matching bits (e.g., 7) resulted in a lower fault injection rate, simulating sporadic fault events, while fewer matching bits (e.g., 3) led to a higher fault injection rate, simulating more frequent incidents. A moderate value, such as 5, balanced fault frequency and system reliability evaluation, allowing for an adequate assessment of the impact of redundancy mechanisms on the ATTNoC system's reliability. Communication within the ATTNoC followed a time-triggered schedule, allowing eight packets to be transmitted within a specific time defined by the schedule. Each period lasted approximately 488.28 μs, and each packet consisted of 16 flits, each composed of 32 bits. The experimental setup involved the exchange of 50,000 packets for evaluation. To assess the reliability of the ATTNoC, three distinct cases were defined: the baseline case without any injected faults, the second case involving the deactivation of redundancy in the safety-critical NI while still injecting faults, and the third case activating the redundancy mechanism in the safety-critical NI and injecting faults using the proposed fault injection IP. Through these experimental cases, the study aimed to gain insights into how redundancy mechanisms enhance the reliability of the ATTNoC, particularly in the presence of transient faults. The evaluation provided valuable information about the system's fault tolerance capabilities, shedding light on the effectiveness of redundancy strategies in critical communication scenarios.

D. Results and Discussion

The objective of the experiment was to evaluate the effects of transient faults that cause message corruption in the ATTNoC and assess the efficiency of redundancy mechanisms to tolerate such faults. The results were analyzed and compared across three cases, as illustrated in Figure 10. This figure



NI		With Redudancy Mechanisms	Without Redundancy Mechanisms
NI (Source)	NI (Sink)	Path	Path
0	1	R7-R4-R1	R7-R4-R1
1	0	R1-R4-R7	R1-R4-R7
2	3	R6-R7-R8/ Redundant (R0-R1-R2))	R6-R7-R8
3	2	R8-R7-R6/ Redundant (R2-R1-R0)	R8-R7-R6
2	1	R0-R1	R0-R1
2	0	R6-R7	R6-R7
3	1	R2-R1	R2-R1
0	2	R7-R6	R7-R6

Fig. 8. Block diagram of ATTNoC with Fault Injection IP (FI-IP).

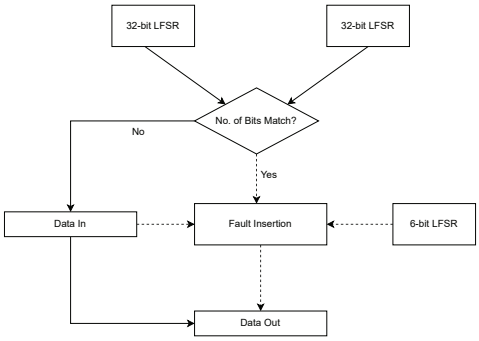


Fig. 9. Flow of Transient Fault Injection

compares the number of packets received without corruption and the error rate for each case. The observations from the analysis are as follows.

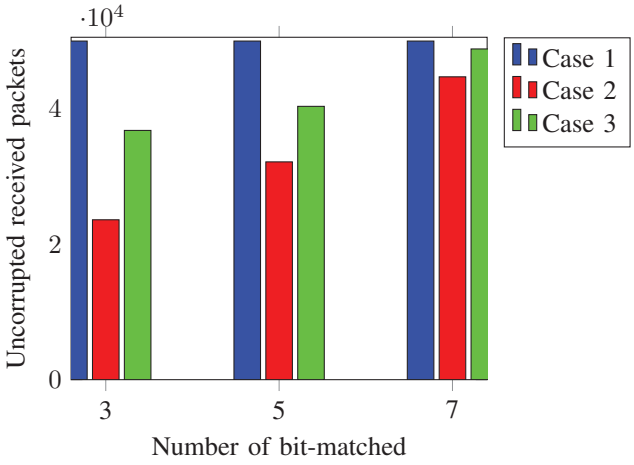


Fig. 10. Comparison of Uncorrupted Packets Received vs Error Rate for the Three Cases

In Case 1 (blue), no faults were injected, and the number of uncorrupted packets remained constant at 50,000, providing a reference for an ideal fault-free scenario. In Case 2 (red), transient faults were injected without redundancy mechanisms, resulting in an increasing number of corrupted packets as the number of matching bits between the two LFSRs decreased. For instance, with three matching bits, approximately 23,611 packets remained uncorrupted, highlighting the impact of

transient faults on system reliability without redundancy. In Case 3 (green), transient faults were injected with redundancy mechanisms, significantly improving the system's fault tolerance even at higher fault injection rates. With three matched bits, about 36,805 packets remained uncorrupted, indicating that the redundancy mechanisms effectively mitigated the effects of transient faults and preserved more uncorrupted packets compared to Case 2. This result highlights the negative impact of transient faults that cause message corruption in the ATTNoC, which reduces the number of uncorrupted packets received at the NI receiver. However, using redundancy mechanisms in the ATTNoC design mitigated this effect, improved fault tolerance, and preserved more uncorrupted packets. It is important to note that only the safety-critical NI uses redundancy, while the non-safety-critical NI lacks redundancy support and requires other fault tolerance mechanisms, such as temporal redundancy using retransmission, to handle message corruption. However, retransmission due to corruption can introduce delays in communication. Minimizing the number of corrupted packets can free up bandwidth, allowing more data transmission and improving overall system performance. Although redundancy mechanisms enhance fault tolerance, they can also introduce additional overhead, such as increased resource usage on the hardware. For example, adding redundancy to time-triggered NIs increased the number of registers and LUTs. Based on the results in Figure 10, the importance of using redundancy mechanisms in safety-critical applications, such as avionics or defense systems, becomes evident [23]. However, for applications where stringent safety requirements are not paramount, like video processing, some transient faults that corrupt messages can be tolerated to some extent. In the case of video data, a transient fault causing corruption in a single pixel does not significantly affect the overall quality of the video. Therefore, extensive redundancy mechanisms or error correction techniques may be less crucial in such scenarios. The decision to use redundancy mechanisms should be based on the specific requirements and criticality of the application. Safety-critical domains require robust fault-tolerant measures to ensure reliable operation and avoid potential hazards. In contrast, for less safety-critical applications, the impact of transient faults on system functionality or output quality may be tolerable, allowing a more flexible approach to implementing redundancy mechanisms.

V. Conclusion and Future Work

The NoC-based multi-core architecture presented in this paper offers a versatile solution to enhance the reliability in embedded systems, particularly in real-time applications. The need for safe applications in embedded systems has been the driving force behind this research. The ATTNoC addresses these challenges by incorporating redundancy and adaptation mechanisms. The NoC's adaptability allows for schedule switching in response to context events, ensuring the isolation of faulty resources and the seamless transfer of tasks or messages to alternative resources. This mechanism guarantees reliable communication and can handle router faults by rerouting messages, thereby maintaining uninterrupted communication.

Moreover, seamless redundancy at the NI level provides dual-channel transmission for critical messages, allowing the NoC to tolerate faults in routers or links during message exchanges. The research extends the functionality of event-triggered NoC architectures, specifically LISNoC [2], by introducing source-based routing, adaptability, seamless redundancy, and time-triggered communication. These extensions enhance its capabilities, making it suitable for real-time systems and adaptable to changing requirements.

In future work, AI-based solutions for event prediction and schedule generation based on this event prediction also hold promise in enhancing the architecture's resource usage without requiring a large amount of memory to store precomputed schedules. Predicting events in advance and storing AI-generated schedules in memory could reduce the time and memory overhead required for schedule switching. This approach could enhance the architecture's efficiency and responsiveness, making it even more effective in meeting the demands of real-time embedded systems.

References

[1] I. Walter, I. Cidon, and A. Kolodny, "Benoc: A bus-enhanced network on-chip for a power efficient cmp," *IEEE Computer Architecture Letters*, vol. 7, no. 2, pp. 61–64, 2008.

[2] TUM, "Lisnoc," https://www.ce.cit.tum.de/lis/forschung/aktuelle-projekte/optimsoc/lis-noc/, accessed 2022-14-09.

[3] R. Obermaisser, H. Ahmadian, A. Maleki, Y. Bebawy, A. Lenz, and B. Sorkhpour, "Adaptive time-triggered multi-core architecture," *Designs*, vol. 3, no. 1, 2019. [Online]. Available: https://www.mdpi.com/2411-9660/3/1/7

[4] E. Dubrova, *Fault-Tolerant Design: An Introduction*. Kluwer Academic Publishers, 2008.

[5] M. Radetzki, C. Feng, X. Zhao, and A. Jantsch, "Methods for fault tolerance in networks-on-chip," *ACM Computing Surveys (CSUR)*, vol. 46, 10 2013.

[6] S. Murali, D. Atienza, L. Benini, and G. De Micheli, "A multi-path routing strategy with guaranteed in-order packet delivery and fault-tolerance for networks on chip," in *2006 43rd ACM/IEEE Design Automation Conference*, 2006, pp. 845–848.

[7] T. Lehtonen, P. Liljeberg, and J. Plosila, "Analysis of forward error correction methods for nanoscale networks-on-chip," 5 2010.

[8] A. Eghbal, P. M. Yaghini, H. Pedram, and H. R. Zarandi, "Designing fault-tolerant network-on-chip router architecture," *International Journal of Electronics*, vol. 97, no. 10, pp. 1181–1192, 2010. [Online]. Available: https://doi.org/10.1080/00207217.2010.512016

[9] M. R. Kakoee, V. Bertacco, and L. Benini, "Relinoc: A reliable network for priority-based on-chip communication," in *2011 Design, Automation Test in Europe*, 2011, pp. 1–6.

[10] K. Constantinides, S. Plaza, J. Blome, B. Zhang, V. Bertacco, S. Mahlke, T. Austin, and M. Orshansky, "Bulletproof: a defect-tolerant cmp switch architecture," in *The Twelfth International Symposium on High-Performance Computer Architecture, 2006.*, 2006, pp. 5–16.

[11] N. Kadri and M. Koudil, "A survey on fault-tolerant application mapping techniques for network-on-chip," *Journal of Systems Architecture*, vol. 92, pp. 39–52, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1383762118301498

[12] Aniket and R. Arunachalam, "A novel algorithm for testing crosstalk induced delay faults in vlsi circuits," in *18th International Conference on VLSI Design held jointly with 4th International Conference on Embedded Systems Design*, 2005, pp. 479–484.

[13] Y. Kim, R. Daly, J. Kim, C. Fallin, J. H. Lee, D. Lee, C. Wilkerson, K. Lai, and O. Mutlu, "Flipping bits in memory without accessing them: An experimental study of dram disturbance errors," in *2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA)*, 2014, pp. 361–372.

[14] Y. Ouyang, Q. Wang, Z. Li, H. Liang, and J. Li, "Fault-tolerant design for data efficient retransmission in winoc," *Tsinghua Science and Technology*, vol. 26, no. 1, pp. 85–94, 2021.
[15] J. Wang, M. Ebrahimi, L. Huang, A. Jantsch, and G. Li, "Design of fault-tolerant and reliable networks-on-chip," in *2015 IEEE Computer Society Annual Symposium on VLSI*, 2015, pp. 545–550.
[16] A. DeOrio, D. Fick, V. Bertacco, D. Sylvester, D. Blaauw, J. Hu, and G. Chen, "A reliable routing architecture and algorithm for nocs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 5, pp. 726–739, 2012.
[17] K. Constantinides, S. Plaza, J. Blome, B. Zhang, V. Bertacco, S. Mahlke, T. Austin, and M. Orshansky, "Bulletproof: a defect-tolerant cmp switch architecture," in *The Twelfth International Symposium on High-Performance Computer Architecture, 2006.*, 2006, pp. 5–16.
[18] B. Motruk, J. Diemer, R. Buchty, and M. Berekovic, "Power monitoring for mixed-criticality on a many-core platform," in *Architecture of Computing Systems – ARCS 2013*, H. Kubátová, C. Hochberger, M. Daněk, and B. Sick, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 13–24.
[19] R. Nambinina, D. Onwuchekwa, S. Nahar, D. Sheladiya, and R. Obermaisser, "Extension of the lisnoc (network-on-chip) with an axi-based network interface," in *Proc. of the 6th International Conference on Computing Methodologies and Communication (ICCMC)*, 2022.
[20] D. Onwuchekwa, J. Paulachan, R. Nambinina, and R. Obermaisser, "Time-triggered network interface extension for the versal network-on-chip," in *2023 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, Feb 2023, pp. 582–589.
[21] A. Lenz, T. Pieper, and R. Obermaisser, "Global adaptation for energy efficiency in multicore architectures," in *2017 25th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, 2017, pp. 551–558.
[22] P. Lala, "Transient and permanent fault injection in vhdl description of digital circuits," in *Circuits and Systems, 2012.*, vol. 3, no. 2, 2012, pp. 192–199.
[23] R. HEARN, "Military embedded systems," *Optimizing avionics reliability with dissimilar redundant architectures*, Dec 06, 2018. [Online]. Available: https://militaryembedded.com/avionics/computers/optimizing-reliability-dissimilar-redundant-architectures

This work has been supported by the research project FRACTAL in part by the EC under grant number 877056 and the BMBF under grant number 16MEE015K.